
CHALMERS

GÖTEBORGS UNIVERSITET

Tillämpning och Visualisering av Kvaternioner

Application and Visualization of Quaternions

Examensarbete för kandidatexamen i matematik vid Göteborgs universitet
Kandidatarbete inom civilingenjörsutbildningen vid Chalmers

Emil Hietanen
Hampus Ahlebrand
Henrik Petterson
Philip Karlsson

Institutionen för Matematiska vetenskaper
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg, Sverige 2020

Tillämpning och Visualisering av Kvaternioner

Examensarbete för kandidatexamen i matematik vid Göteborgs universitet
Emil Hietanen

Kandidatarbete i matematik inom civilingenjörsprogrammet Teknisk fysik vid Chalmers
Hampus Ahlebrand

Kandidatarbete i matematik inom civilingenjörsprogrammet Teknisk matematik vid Chalmers
Henrik Petterson

Kandidatarbete i matematik inom civilingenjörsprogrammet Kemiteknik och fysik vid Chalmers
Philip Karlsson

Handledare: Alexei Heintz
Exminatorer: Ulla Dinger & Maria Roginskaya

Institutionen för Matematiska vetenskaper
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg, Sverige 2020

Förord

Vi vill tacka vår handledare Alexei Heintz och även våra eximatorer: Ulla Dinger och Maria Roginskaya.

En loggbok (dagbok) har uppdaterats varje vecka inom gruppen. Individuella tidsloggar har förts där information om de enskilda medverkandets prestationer återfinns.

Gemensamma bidrag

Vi har alla hjälpt varandra genom att noggrant läsa igenom varandras bidrag och givit feedback.

Individuella bidrag

Avsnitt	Emil	Hampus	Henrik	Philip
Poulärvetenskaplig presentation	x			
Sammanfattning	x			
Inledning		x		
Förkunskap		x		
Syfte		x		
Historia	x			
Grundläggande Teori och Notation				
-Notation		x		
-Räkneregler och Defentioner		x		
-Rotation		x		
-Visualisering			x	
Tillämpningar av Grundteori				
-Bält-tricket				x
-Boll-tricket		x	x	
Avancerad teori				
-3D-ytor	x			
-Visualisering av 3D-ytor	x			
Tillämpning av Avancerad Teori				
-Kollisionssdetekion	x			
-Kollision med friktion		x		
-Visualisering av en kollision	x			x
Diskussion	x			
Billagor				
-A Kollsion bevis		x		
-B Superquadratics	x			
-C Kollisionsdetektion och restriktione	x			x
-D Visualiseringar				
-D.1 Blender	x			x
-D.2 MatLab	x	x	x	
Tabellutforming	x	x		

Figur 1: En representation av ansvarig/ansvariga för respektive avsnitt.

Populärvetenskaplig presentation

Tillämpningar av Kvaternioner

Sir William Rowan Hamilton var enligt egen utsago ute på en promenad år 1854 när han fick idén om hur de komplexa talen kan generaliseras upp till fyra dimensioner, denna upptäckt blev senare känd som den hyperkomplexa talmängden kvaternioner. Hamilton föreslog att man bör använda sig av kvaternioner som standard notation för vektoroperationer men förslaget mötte motstånd från forskningsvärlden och istället valdes Gibbs notation. Möjligheten att använda sig av kvaternioner diskuterades även i samband med tillkomsten av kvantfysiken men återigen valdes en annan metod. En kan undra varför kvaternioner dömdes ut till förmån för alternativa metoder, det främsta argumentet var att notationen ansågs vara otymplig.

I början av 80-talet i samband med datorgrafikens framfart så ökade också användandet av kvaternioner eftersom det inom datorgrafiken var (och är) viktigt med tillförlitlig orientering vid 3D-rotationer. Användandet av kvaternioner har därför varit att föredra framför den alternativa metoden, Eulervinklar. Anledningen till att kvaternioner ofta valts framför Eulervinklar är på grund av Gimbal-lock. Gimbal-lock är ett fenomen parat med dessa vinklar som innebär att en rotationsaxel förloras vid en specifik orientering, ett system i denna situation mister därför kontakten med en mängd orienteringar. En annan fördel med att nyttja kvaternioner är möjligheten att förklara och förstå fenomen som annars inte avslöjar sina hemligheter. För att demonstrera detta har vi i rapporten visualiserat två sådana fenomen, Bält- och Boll-tricket. Problematiken och begränsningarna med Eulervinklar har inneburit att man inom flera områden valt att använda sig av kvaternioner som standardmetod för rotation och orientering.

Inom stelkroppsdynamiken är det viktigt att beskriva hur stelkroppar påverkas av en kollision, där en stelkropp är en kropp som inte kan deformeras. För närvarande finns inget känt material med denna egenskap, men en stelkropp är en bra approximation om deformationen är försumbar. I samband med en kollision av två stelkroppar så kommer deras orientering och rotation att ändras, en kollisionsmodell behöver därmed ett verktyg för att beskriva dessa ting vilket fördelaktigt görs med kvaternioner. För att visa hur det kan göras har vi i denna rapport simulerat och visualiserat två kollisioner med friktion. Ytterligare ett användningsområde för kvaternioner är inom differentialgeometrin där kvaternioner bland annat kan användas för att beskriva 3D-ytor, dess egenskaper och avbildning.

-Vi behöver använda kvaternioner varje gång vi i vill interpolera koordinater, animation av en kameras rörelse, simuleringar, beskriva hastigheten för vätskor, och det är enkelt att göra fel inom allt detta, säger John C. Hart, professor i datorvetenskap [1]. Kvaternioner har flera tillämpningsområden och vi behöver använda oss av kvaternioner, inom alla moderna vetenskaper, för att beskriva den verkliga världen vi lever i.

[1] Andrew J. Hanson. Visualizing Quaternions, 2006. Sid. xxiii

Sammanfattning

Den här rapporten undersöker hur kvaternioner kan visualiseras och användas i diverse tillämpningar. Generaliseringen av komplexa tal upp till fyra dimensioner, senare känt som den hyperkompelxa talmängden kvaternioner presenterades för världen av Sir William Rowan Hamilton, 1854, och har sedan dess applicerats inom flera områden. I denna rapporten utreds hur kvaternioner kan användas och visualiseras inom tre tillämpningsområden men först beskrivs nödvändig teori och bakgrund. I rapporten har en jämförelse mellan kvaternioner och den alternativa metoden, Eulervinklar, studerats för att belysa skillnader i metoderna. Den grundläggande teorin för kvaternioner används därefter för att få en förståelse för teorin och hur kvaternioner fungerar genom två visualiserade fenomen vars förklaring endast är möjlig med kvaternioner, boll- och bält-tricket. Därefter behandlas mer avancerad teori inom tre tillämpningsområden; datorgrafik, stelkroppsdynamik och differentialgeometri.

Inom differentialgeometri så behandlar denna rapporten hur man kan beskriva 3D ytor med hjälp av så kallade kvaternionramar och kvaternion-Gaussavbildning, vilket möjliggör ett sätt att beskriva en yta och dess egenskaper. Dessutom undersöks möjligheterna att använda kvaternioner inom stelkroppsdynamik genom en simulerad kollision mellan två konvexa stelkroppar med en friktionskoefficient och varför det kan vara fördelaktigt att använda sig av kvaternioner. Visualiserngarna av kvaternioner i rapporten är genomförda med olika metoder, men alla visualiseringar har genomförts i antigen Blender eller MatLab. Resultatet i rapporten ger en kännedom om diverse tillämpningar och visualiseringar om kvaternioner. Dessutom belyses också begränsningar med den alternativa metoden, Eulervinklar, som kvaternioner inte besitter. De begränsningar som Eulervinklar besitter innebär att kvaternioner är en fördelaktig metod i de tillämpningar vi studerat.

Nyckelord; Kvaternion; Kvatternionram; Gaussavbildning; Rotation; Orientering; Stelkroppsdynamik; Visualisering, Bält-tricket; Bolltricket.

Abstract

This report examines how quaternions can be used and visualized in various applications. The generalization of complex numbers up to four dimensions, later known as the hypercomplex number of quaternions was presented to the world by Sir William Rowan Hamilton, 1854, and has since been applied in several fields. This report investigates how quaternions can be used and visualized in three areas of application, but first the necessary theory and background are described. In the report, a comparison between quaternions and the alternative method, Euler angles, has been studied to elucidate differences in the methods. The basic theory of quaternions is then used to gain an understanding of the theory and how quaternions work through two visualized phenomena that we explain using quaternions, the ball and belt trick. Thereafter, more advanced theory is dealt with in three areas of application; computer graphics, body dynamics and differential geometry

In differential geometry, this report deals with how to describe 3D surfaces using so-called quaternion frames and quaternion Gaussian map, which allows a way to describe a surface and its properties. In addition, the possibilities of using quaternions in rigid body dynamics are investigated through a simulated collision between two convex rigid bodies with a coefficient of friction and why it can be advantageous to use quaternions. The visualizations of quaternions in the report are carried out by different methods, but all visualizations have been performed in either Blender or MatLab. The results in the report provide knowledge of various applications and visualizations of quaternions. In addition, limitations are also highlighted by the alternative method, Euler angles, which quaternions do not possess. The limitations that Euler angles possess mean that quaternions are an advantageous method in the applications we studied.

Keywords; Quaternion; Quaternion frame; Gauss map; Rotation; Orientation; Rigid-body dynamics; Visualization, Belt trick; Ball trick.

Innehåll

Bilagor

1 Inledning

Detta arbete utreder hur kvaternioner kan visualiseras och tillämpas. Kvaternionen är dock okänd för de flesta trots dess runt 170 år långa historia och att den idag tillämpas inom vida områden som datorgrafik, stelkroppsdynamik, signalbehandling, differentialgeometri, reglerteknik med mera. Kvaternionen har också gjort sig påmind inom kvantfysiken och noterbart är att James Clerk Maxwell först formulerade den klassiska elektromagnetismen med kvaternioner [1] [2]. Inget av de nämnda fälten är obskyrt eller saknar intressenter, snarare gäller det motsatta. Detta leder till åtminstone två frågor; Varför är kvaternionen så okänd? och Varför så användbar? Den förra besvaras kortfattat i kapitlet 4 om kvaternionens historia och den senare frågan har många svar eftersom kvaternionen har en mängd användbara och intressanta egenskaper. Denna rapport ämnar utreda några av dessa.

En egenskap som tidigt upptäcktes var att kvaternionen kunde användas för att representera och visualisera orientering. Det är en nyttig egenskap som delvis förklarar dess användning i ovannämnda områden. Vi redogör för teorin som visar att kvaternionen har dessa egenskaper och även hur de kan användas för att förklara två företeelser som i rapporten benämns Boll- och Bält-Tricket.

Inom tillämpningar används ofta alternativa metoder för att beskriva orientering och två sådana teoretiska verktyg är axel-vinkel framställningen och Eulers vinklar α, β och γ. Den förra beskriver kring vilken axel och vinkel ett referenssystem behöver roteras för avsedd orientering. Vardera Eulervinkel motsvarar rotationsvinkeln kring en basaxel i ett ortogonalt system. Kvaternionframställningen har två viktiga fördelar relativt båda metoderna (för än mer nyanserad jämförelse se kapitel 5.3). Kvaternionen kan tolkas som en funktion vars definitionsmängd utgörs av ramar (för definition se kapitel 5.2) och dess värdemängd av fyrdimensionella punkter vilket för det första tillåter entydig beskrivning av distansen/likheten mellan ramar och för det andra möjliggörs visualisering av rotationssekvenser [2]. En anledningen till att Boll- och Bält-Tricket studeras är för att visa på nyttan av dessa fördelar.

Ytterligare en aspekt som utreds är kvaternionens förmåga att beskriva 3D-ytor. Detta är viktigt inom bl.a differentialgeometri och datorgrafik. Speciellt undersöks hur kvaternionramar och kvaternion-Gaussavbildning relaterar till egenskaper hos 3D-ytor.

För att visa på ytterligare tillämpning av den presenterade teorin så avslutas rapporten med två kollisionssimuleringar. En del av den mer avancerade teorin appliceras i en algoritm för kollisionsdetektion och kropparnas orientering beräknas och visualiseras med hjälp av den grundläggande teorin, och programmet Blender som animerar kollisionen tillämpar dessutom kvaternionteori. Notera att kollisionssimuleringen i sig är inte är huvudintresset men för ett någorlunda praktiskt resultat har vi valt att inkludera friktion i kollisionsmodellen vilket ger verklighetstrogna resultat [3].

Avslutningsvis påpekas att detta arbete är en litteraturstudie mestadels influerad av boken *Visualizing Quaternions* av Andrew J. Hanson. Från denna text kommer stora delar av den presenterade teorin och ofta används samma notation. Texten rekommenderas för den som vill fördjupa sig i ämnet.

2 Förkunskap

I rapportens första hälft behandlas grundläggande kvaternionteori och för att förstå denna är det en fördel att ha grundkunskaper inom linjär algebra, komplexa tal och flervariabelanalys. I rapportens andra hälft utreds mer avancerad teori och där är det nyttigt med grundläggande kunskaper inom differentialgeometri. För den som inte är bekant med differentialgeometri så rekommenderas läsaren att först läsa kapitel 19 och 20 i *Visualizing Quaternions* av Andrew J. Hanson.

3 Syfte

Syftet är att visa hur kvaternioner och dess visualisering kan tillämpas. De tillämpningar som granskas är rotationsberäkning, Bält-tricket, Boll-tricket, beskrivning av 3D-ytor och kollisionssimulering.

4 Historia

Vi kommer i detta avsnitt ge lite historik kring kvaternioner för att läsaren skall få en bakgrund till arbetet.

Sir William Rowan Hamilton undersökte under början av 1800-talet möjligheterna att utvidga de komplexa talen. De komplexa talen består av en reell och en komplex del, vilket beskrivs i formen $a + bi$ där a, b tillhör de reella talen och där talet i är den imaginära delen. Hamilton hade arbetat med att generalisera de komplexa talen under några år och under en promenad med sin fru året 1854 fick han plötsligt idén att man kan genom att använda sig av regeln $i^2 = j^2 = k^2 = ijk = -1$ utvidga de komplexa talen till fyra dimensioner, denna utvidgning är det vi kallar för kvanternioner. I samband med Hamiltons upptäckt så ställdes frågan om kvaternioner generaliserar komplexa tal, kan kvaternioner generaliseras ytterligare. Det visade sig vara möjligt att generalisera kvaternioner till åtta dimensioner, därav namnet octanioner eller dual-kvaternioner som de också benämns. Dual-kvaternioner kan beskrivas precis som kvaternioner men man använder sig av ett ordnat par $\hat{a} = (a, b)$ istället som koefficienter. Det har sedan dess skett fler generaliseringar av kvaternioner men dual-kvaternioner är de mest användbara [2].

Hamilton förespråkade användandet av kvaternioner som standard notation för att beskriva tredimensionella vektoroperationer men kvaternioner ansågs vara en obekväm notation. Istället valdes Gibbs notation $\mathbf{x} = (x, y, z)$ som standardnotation för att beskriva en tredimensionell vektor. När kvantteorin för elektroner utvecklades under tjugohundratalet så blev det uppenbart att kvaternioner hade ett samband med spinorer (se kapitel 6.1) men istället valdes den alternativa notationen i form av en 2x2 matris, känd som Pauli matriser. Tidigt användes kvaternioner inom Aeronautik och astronautik men det blev framförallt populärt i samband med utvecklingen av tredimensionell datorgrafiken. Inom datorgrafiken behövde man vara precis med orienteringen för tredimensionella objekt och kameror, då insåg man problematiken med att använda sig av Eulervinklar (se kapitel 5.3.3). Problematiken med Eulervinklar innebar att man istället använde sig av kvaternioner som standardmetod för att beskriva orientering och rotation. Det ökade användandet av kvaternioner inom datorgrafiken innebar också att nyttjandet av kvaternioner ökade inom diverse tillämpningsområden [2].

5 Grundläggande Teori och Notation

I detta kapitel redogörs teori och notation som används genom hela rapporten. Läsaren rekommenderas att bekanta sig med notationen, detta för att undvika frågor som; Är det en vektor eller en skalär? Därefter följer ett avsnitt med några nyttiga definitioner och operationer relaterade till kvaternioner. I avsnitt 5.3 brukas dessa verktyg för att visa hur kvaternioner kan beskriva rotation och orientering. Det avslutande avsnitten ägnas åt metoder för att visualisera kvaternioner.

5.1 Notation

Här ges den generella notationen som används. På ställen där specifik eller annan notation krävs så kommer det först att klargöras, detta gäller framförallt för kapitel 7 och 8. Men allmänt gäller nedanstående tabell.

Allmän Notation	
Skalär	Skrivs kursivt, t.ex. a men det finns undantag. De imaginära enheterna benämns i, j och k och kvaternioner tecknas också kursivt, se nedan.
Kvaternion	Skrivs kursivt, speciellt är q reserverad men andra tecken kan användas som t.ex. p. Från kontext bör det vara enkelt att urskilja då en kvaternion eller skalär åsyftas.
Vektor	Fetstilad minuskel, t.ex. $\boldsymbol{v}$
Enhetsvektor	Identifieras med en hatt t.ex. $\hat{\boldsymbol{n}}$
Matris	Kursiv versal, t.ex. R

5.2 Räkneregler och Definitioner

En kvaternion q består av en reell och tre imaginära komponenter. Explicit gäller alltså

$$q = q_0 + iq_1 + jq_2 + kq_3 \tag{1}$$

där $q_i \in \mathbb{R}$, $i \in 0, 1, 2, 3$ där

$$\begin{gathered} i^2 = j^2 = k^2 = -1 \\ ij = k, jk = i, ki = j \\ ji = -k, kj = -i, ik = -j \end{gathered} \tag{2}$$

[4]. Notera förhållandet mellan de imaginära enheterna i, j och k, de beter sig likt tre ortogonala enhetsvektorer. Detta är en nyttig egenskap eftersom en viss formalism tillåter implicit behandling av reglerna (2). Låt kvaternionen tecknas som en 4-vektor $q = [q_0, q_1, q_2, q_3]$, eller $q = [q_0, \mathbf{q}]$, där $\mathbf{q}$ är en 3-vektor (innehåller de imaginära komponenterna). Då kan operationerna kvaternionmultiplikation, -skalärprodukt och -konjungering i respektive ordning skrivas som nedan i ekvationerna (3), (4) och (5). Dessa bevisas enkelt med ekvation (1) och sambanden i (2) [2] [4] [5].

Multiplikation:

$$p \star q = [p_0, p_1, p_2, p_3] \star [q_0, q_1, q_2, q_3] = [p_0 q_0 - \mathbf{p} \cdot \mathbf{q}, p_0\mathbf{q} + q_0\mathbf{p} + \mathbf{p} \times \mathbf{q}] \tag{3}$$

Skalärprodukt:

$$p \cdot q = [p_0, p_1, p_2, p_3] \cdot [q_0, q_1, q_2, q_3] = p_0 q_0 + \mathbf{p} \cdot \mathbf{q} \tag{4}$$

Konjungering:

$$\bar{q} = [q_0, -q_1, -q_2, -q_3] = [q_0, -\mathbf{q}] \tag{5}$$

Vid kvaternionmultiplikation utelämnas hädanefter $\star$ operatorn. För våra ändamål kommer det visa sig att endast normaliserade kvaternioner behöver betänkas (se avsnitt 5.3.1). I fortsättningen antas därmed att

$$q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1. \tag{6}$$

Notera att talen q_0, q_1, q_2 och q_3 under begränsningen (6) beskriver ytan till en hypersfär vilket kommer utnyttjas för visualisering, se kapitel 5.4. Restriktionen leder även till att kvaternioninversen ges av dess konjungat [4].

Sats 1. $q\bar{q} = [1, 0] = \bar{q}q$

Bevis. $q\bar{q} = [q_0 q_0 + \mathbf{q} \cdot \mathbf{q}, -q_0\mathbf{q} + q_0\mathbf{q} - \mathbf{q} \times \mathbf{q}] = [q_0^2 + q_1^2 + q_2^2 + q_3^2, 0] = [1, 0]$ □

Nu följer några nyttiga definitioner och Eulers fundamentala sats om rotationer

Definition 1. *En **enhetskvaternion** är en normaliserad kvaternion.*

Definition 2. *Mängden $\mathbb{S}^d$ där $d \in \mathbb{N}$ innehåller ytpunkterna till en $d+1$-dimensionell sfär.*

Definition 3. *En **rotation** definieras av rotationsaxel givet av normalvektor $\hat{\boldsymbol{n}}$ och en rotationsvinkel θ sådana att en vektor roterar kring $\hat{\boldsymbol{n}}$ med den givna vinkeln θ.*

Definition 4. *En **ram** (eller **orientering**) definieras av tre inbördes ortogonala normalvektorer, säg $\hat{\boldsymbol{x}}$, $\hat{\boldsymbol{y}}$ och $\hat{\boldsymbol{z}}$. Om en rotation appliceras så fås en annan ram.*

Definition 5. *En **kvaternionram** är en punkt $q \in \mathbf{S}^3$.*

Sats 2. *Givet två ramar R_1 och R_2 så existerar en rotation sådan att R_1 transformeras till R_2.*

Ofta definieras orientering av en normalvektor men i denna rapport särskiljs inte ram och orientering [5]. Redan nu noteras att kvaternionramar (detta namn eftersom en kvaternion kan motsvara en ram) används för att visualisera rotationssekvenser som kurvor i $\mathbf{S}^3$, se kapitel 6 och 8.3. Till sist, innan vi går vidare definieras rena kvaternioner.

Definition 6. *Kvaternionen vars reella komponent är noll är en **ren kvaternion** [1].*

5.3 Rotation

I inledningen och senare har det nämnts att kvaternioner har en förmåga att beskriva rotation och orientering och i detta avsnitt visas att så är fallet. Initialt granskas axel-vinkel-framställningen och hur den kan beskrivas av en kvaternion, samt en tolkning av kvaternionmultiplikation. Därefter härleds en kvaternionoperator som roterar rena kvaternioner följd av en redogörelse för Eulers vinklar och hur fenomenet Gimbal-lock uppträder. Avsnittet avslutas med en icke-rigorös jämförelse mellan de tre metoderna.

5.3.1 Rotation med kvaternioner i matrisform

Om en kvaternion kan beskriva rotation bör den kunna relateras till en rotationsmatris därför reder vi först ut hur en vektor $\boldsymbol{v} = [v_1, v_2, v_3]$ kan roteras kring en godtycklig axel $\hat{\boldsymbol{n}} = [n_1, n_2, n_3]$. Ett svar ges av axel-vinkel-modellen där rotationsaxeln och -vinkeln explicit anges, dess matrisform ges i nedanstående sats och bevis.

Sats 3. *Låt $R(\theta, \hat{\boldsymbol{n}}) \in 3 \times 3$ vara en rotationsmatris med rotationsaxeln $\hat{\boldsymbol{n}}$ och rotationsvinkeln θ. $R(\theta, \hat{\boldsymbol{n}})$ har då formen angiven av ekvation (7).*

$$R(\theta, \hat{n}) = \begin{bmatrix} c + n_1^2(1-c) & n_1 n_2(1-c) - s n_3 & n_1 n_3(1-c) + s n_2 \\ n_2 n_1(1-c) + s n_3 & c + n_2^2(1-c) & n_2 n_3(1-c) - s n_1 \\ n_3 n_1(1-c) - s n_2 & n_3 n_2(1-c) + s n_1 & c + n_3^2(1-c) \end{bmatrix} \tag{7}$$

där $c = \cos\theta$ och $s = \sin\theta$ [2].

Bevis. Vi antar att de elementära rotationsmatriserna (8) är kända:

$$R(\theta, \hat{\boldsymbol{x}}) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}, \; R(\theta, \hat{\boldsymbol{y}}) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}, \; R(\theta, \hat{\boldsymbol{z}}) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{8}$$

En sekvens av ovanstående matriser söks, sådan att den önskade rotationen åstadkoms. För att finna en lämplig sekvens parametriseras $\hat{\boldsymbol{n}}$ med sfäriska koordinater: $\hat{\boldsymbol{n}} = [\cos\alpha \sin\beta, \sin\alpha \sin\beta, \cos\beta]$. Låt $\hat{\boldsymbol{n}}'$ beteckna en rotation av $\hat{\boldsymbol{n}}$. Strategin är att genomföra ett basbyte sådant att ett av det nya systemets basaxlar är $\hat{\boldsymbol{n}}$ och därefter applicera en lämplig elementär rotationsmatris och sedan invertera basbytet. Detta är ekvivalent med att först rotera $\hat{\boldsymbol{n}}$ så $\hat{\boldsymbol{n}}' = \hat{\boldsymbol{z}}$ och därefter applicera $R(\theta, \hat{\boldsymbol{z}})$ och sedan invertera initiala rotationen så $\hat{\boldsymbol{n}}' = \hat{\boldsymbol{n}}$. Låt nu $\boldsymbol{v}'$ beteckna rotationen av $\boldsymbol{v}$ kring $\hat{\boldsymbol{n}}$ med vinkeln θ, från argumentet ovan och parametriseringen inses då att $\boldsymbol{v}' = R(\alpha, \hat{\boldsymbol{z}})R(\beta, \hat{\boldsymbol{y}})R(\theta, \hat{\boldsymbol{z}})R(-\beta, \hat{\boldsymbol{y}})R(-\alpha, \hat{\boldsymbol{z}})\boldsymbol{v}$. Och således

$$R(\theta,\hat{\boldsymbol{n}}) = R(\alpha,\hat{\boldsymbol{z}})R(\beta,\hat{\boldsymbol{y}})R(\theta,\hat{\boldsymbol{z}})R(-\beta,\hat{\boldsymbol{y}})R(-\alpha,\hat{\boldsymbol{z}})$$
$$= \begin{bmatrix} c+n_1^2(1-c) & n_1n_2(1-c)-sn_3 & n_1n_3(1-c)+sn_2 \\ n_2n_1(1-c)+sn_3 & c+n_2^2(1-c) & n_2n_3(1-c)-sn_1 \\ n_3n_1(1-c)-sn_2 & n_3n_2(1-c)+sn_1 & c+n_3^2(1-c) \end{bmatrix}.$$

□

För att visa att matrisen (7) beskriver en kvaternion används två variabelbyten. Dessa byten kan tyckas vara obefogade men i nästa avsnitt 5.3.2 ges en motivering och för än mer bakgrund hänvisas läsaren till texterna [2] [4]. Först, låt $a = \cos\frac{\theta}{2}$ och $b = \sin\frac{\theta}{2}$ och notera $c = a^2 - b^2$, $s = 2ab$ samt $1 - c = a^2 + b^2 - (a^2 - b^2) = 2b^2$ och resultatet blir

$$R(\theta,\hat{\boldsymbol{n}}) = \begin{bmatrix} a^2+b^2(n_1^2-n_2^2-n_3^2) & 2b^2n_1n_2-2abn_3 & 2b^2n_3n_1+2abn_2 \\ 2b^2n_1n_2+2abn_3 & a^2+b^2(n_2^2-n_3^2-n_1^2) & 2b^2n_2n_3-2abn_1 \\ 2b^2n_3n_1-2abn_2 & 2b^2n_2n_3+2abn_1 & a^2+b^2(n_3^2-n_1^2-n_2^2) \end{bmatrix}.$$

Matrisen ovan kan i sin tur parametriseras med fyra variabler, q_0, q_1, q_2 och q_3:

$$q_0 = a = \cos\frac{\theta}{2},\ q_1 = n_1b = n_1\sin\frac{\theta}{2},\ q_2 = n_2b = n_2\sin\frac{\theta}{2},\ q_3 = n_3b = n_3\sin\frac{\theta}{2}.$$

Parametriseringen ovan definierar en enhetskvaternion, ty
$q_0^2 + q_1^2 + q_2^2 + q_3^2 = \cos^2\frac{\theta}{2} + (n_1^2 + n_2^2 + n_3^2)sin^2(\theta/2) = 1$ [2]. Påståendet om enhetskvaternioner i avsnitt 5.2 är därmed motiverat. Enhetskvaternionen

$$q = [\cos\frac{\theta}{2}, \sin\frac{\theta}{2}\hat{\boldsymbol{n}}] \tag{9}$$

representerar därmed rotationsmatrisen $R(\theta,\hat{\boldsymbol{n}})$ och beskriver därför en ram vilken alstras genom rotation av referensramen kring $\hat{\boldsymbol{n}}$ med en vinkel $0 \leq \theta \leq 4\pi$, upp till 4π för att kunna representera alla rotationer. Givet en enhetskvaternion kan alltså motsvarande rotationsmatris $R(q)$ skrivas som

$$R(q) = \begin{bmatrix} q_0^2+q_1^2-q_2^2-q_3^2 & 2q_1q_2-2q_0q_3 & 2q_1q_3+2q_0q_2 \\ 2q_1q_2+2q_0q_3 & q_0^2-q_1^2+q_2^2-q_3^2 & 2q_2q_3-2q_0q_1 \\ 2q_1q_3-2q_0q_2 & 2q_2q_3+2q_0q_1 & q_0^2-q_1^2-q_2^2+q_3^2 \end{bmatrix}. \tag{10}$$

Notera att q och $-q$ motsvarar samma matris vilket innebär att avbildningen är bilinjär, detta är en nackdel hos matrisrepresentationen, den tillåter skenbar likhet mellan två olika ramar (se kapitel 6.1).

Nu undersöks närmare vad som sker för två fall av kvaternionmultiplikation, först produkten av två godtyckliga enhetskvaternioner sen produkten av (9) och en ren kvaternion $q_v = [0, \boldsymbol{v}]$.

Låt nu t parametrisera en rotationssekvens längs x-axeln vilket innebär att sekvensen beskrivs av matrisen $R(\theta t,\hat{\boldsymbol{x}})$ och kvaternionen $q = [\cos\frac{\theta t}{2}, \sin\frac{\theta t}{2}\hat{x}]$.
Notera nu att sekvensen kan skrivas som matrismultiplikation:
$R(\theta t,\hat{\boldsymbol{x}}) = R(\theta t_n,\hat{\boldsymbol{x}})R(\theta t_{n-1},\hat{\boldsymbol{x}})...R(\theta t_1,\hat{\boldsymbol{x}})$ (där $\sum_{i=1}^{n} t_i = t$). Det visar sig att $q(t)$ på samma vis kan beskrivas av kvaternionmultiplikation: $q(t) = q(t_n)...q(t_1)$. Slutsatsen är att multiplikationen av två kvaternioner $q_1(\theta_1,\hat{\boldsymbol{n}}_1)$, $q_2(\theta_2,\hat{\boldsymbol{n}}_2)$ motsvarar matrisprodukten av $R_1(\theta_1,\hat{\boldsymbol{n}}_1)$ och $R_2(\theta_2,\hat{\boldsymbol{n}}_2)$ [2]. Operationen har en geometrisk tolkning som utreds i kapitel 6.2.

Önskvärt vore att $q_{v'} = qq_v = [0, R(\theta,\hat{\boldsymbol{n}})\boldsymbol{v}]$, vilket skulle förenkla rotationsberäkningar men räkneexemplet nedan visar att detta inte är fallet [4].
Exempel: Låt $\theta = \frac{\pi}{2}$, $\hat{\boldsymbol{n}} = \frac{1}{\sqrt{2}}[1,1,0]$ och $q_v = [0,2,0,0]$.

$$q_{v'} = qq_v = [-1, \sqrt{2}, 0, -1]$$

Operationen resulterar inte i en ren kvaternion, vektordelen har inte roterats 45 grader kring $\hat{\boldsymbol{n}}$ och inte heller är normen bevarad (hos imaginärdelen). En slutsats från ovanstående är att

enhetskvaternionen q avbildar en ram given av en rotationsmatris till en punkt (q_0, q_1, q_2, q_3). I kvaternionrummet svarar då multiplikation mot ramrotation men inte mot vektorrotation. Härnäst utreds hur kvaternionmultiplikation kan användas för rotationsberäkningar av vektorer.

5.3.2 Rotation av en ren kvaternion

I detta avsnitt söks en rotationsoperator som agerar på rena kvaternioner. Först preciseras två egenskaper en rotation kring en viss axel uppfyller. Låt $\boldsymbol{v}' = f_p(\boldsymbol{v})$ där f_p är en rotationsoperator vilken roterar $\boldsymbol{v}$ kring en axel $\hat{\boldsymbol{p}}$. För en rotation gäller då

$$|\boldsymbol{v}| = |\boldsymbol{v}'| \tag{11}$$

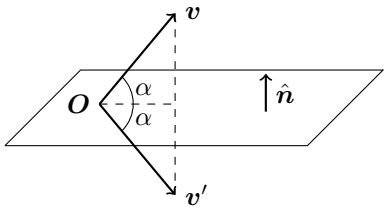
$$(\boldsymbol{v} - \boldsymbol{v}') \cdot \hat{\boldsymbol{p}} = 0 \tag{12}$$

Rotationsoperatorn är normbevarande och rotationsvektorn $(\boldsymbol{v} - \boldsymbol{v}')$ ligger i rotationsplanet. En insikt vi kommer använda är att reflektioner är rotationer vilket nedan bevisas [1].

Sats 4. *Låt $\hat{\boldsymbol{n}}$ vara normalen till ett plan som innehåller origo. Låt $\boldsymbol{v}$ vara en godtycklig vektor så $dim(\boldsymbol{v}) = dim(\hat{\boldsymbol{n}})$. Vidare, låt α vara vinkeln mellan planet och vektorn $\boldsymbol{v}$. För reflektionen $\boldsymbol{v}'$ gäller*

$$\boldsymbol{v}' = \boldsymbol{v} - 2\hat{\boldsymbol{n}}(\hat{\boldsymbol{n}} \cdot \boldsymbol{v}) \tag{13}$$

vilket är en rotation med rotationsvinklen 2α.



Figur 2: Ett plan med normal $\hat{\boldsymbol{n}}$ som innehåller origo. Vi ser också vinkeln α mellan planet och vektorn $\boldsymbol{v}$. Vektorn $\boldsymbol{v}'$ är reflektionen av $\boldsymbol{v}$ m.a.p planet.

Bevis. Från figur 2 inses direkt att att rotationsvinkeln är 2α och

$$\boldsymbol{v}' = \boldsymbol{v} - 2\hat{\boldsymbol{n}}(\hat{\boldsymbol{n}} \cdot \boldsymbol{v}).$$

Vidare gäller

$$\boldsymbol{v}' \cdot \boldsymbol{v}' = |\boldsymbol{v}'|^2 = |\boldsymbol{v}|^2 - 4\cos\alpha + 4\cos\alpha = |\boldsymbol{v}|^2 \Rightarrow |\boldsymbol{v}| = |\boldsymbol{v}'|$$

$$(\boldsymbol{v} - \boldsymbol{v}') \cdot \hat{\boldsymbol{p}} = (\boldsymbol{v} - \boldsymbol{v}') \cdot (\hat{\boldsymbol{n}} \times \hat{\boldsymbol{n}}) = 2\hat{\boldsymbol{n}}(\hat{\boldsymbol{n}} \cdot \boldsymbol{v}) \cdot (\hat{\boldsymbol{n}} \times \hat{\boldsymbol{v}}) = 0.$$

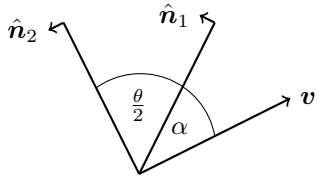
□

Den sökta rotationsoperatorn kan möjligen härledas från en reflektion men för en given rotation måste ett lämpligt plan väljas (implicit i slutändan) vilket gör härledning relativt komplicerad. Om istället reflektionssekvenser studeras undgås komplikationen.

Sats 5. *Låt $\hat{\boldsymbol{n}}_1$ och $\hat{\boldsymbol{n}}_2$ definiera två plan genom origo och låt $\hat{\boldsymbol{p}}$ ge skärningen mellan planen: $\hat{\boldsymbol{n}}_1 \times \hat{\boldsymbol{n}}_2 = \sin\frac{\theta}{2}\hat{\boldsymbol{p}}$ där $\frac{\theta}{2}$ är vinkeln mellan planen: $\hat{\boldsymbol{n}}_1 \cdot \hat{\boldsymbol{n}}_2 = \cos\frac{\theta}{2}$. Om $q_1 = [0, \hat{\boldsymbol{n}}_1]$, $q_2 = [0, \hat{\boldsymbol{n}}_2]$ och $q = q_2 q_1$ samt $q_v = [0, \boldsymbol{v}]$ så gäller att operatorn f:*

$$f(q_v) = q q_v q^{-1}$$

är en rotationsoperator med axeln $\hat{\boldsymbol{p}}$ och rotationsvinkeln θ [1].



Figur 3: Vinkeln α representerar vinkeln mellan plan 1 (med normal $\hat{\boldsymbol{n}}_1$) och vektorn $\boldsymbol{v}$ som ska roteras genom att först reflekteras i plan 1 och sedan i plan 2 (med normal $\hat{\boldsymbol{n}}_2$). Vinkeln mellan planen är $\frac{\theta}{2}$.

Bevis. Direkt ur figur 3 och sats 4 inses att reflektionssekvensen ger en rotationsvinkeln $2\alpha + 2(\frac{\theta}{2} - \alpha) = \theta$. Härnäst studeras produkten $q_1 q_v q_1$.

$$q_1 q_v q_1 = [0, \hat{\boldsymbol{n}}_1][-\boldsymbol{v} \cdot \hat{\boldsymbol{n}}_1, \boldsymbol{v} \times \hat{\boldsymbol{n}}_1] = [-\hat{\boldsymbol{n}}_1 \cdot (\boldsymbol{v} \times \hat{\boldsymbol{n}}_1), (-\boldsymbol{n} \cdot \hat{\boldsymbol{n}}_1)\hat{\boldsymbol{n}}_1 + \hat{\boldsymbol{n}}_1 \times \boldsymbol{v} \times \hat{\boldsymbol{n}}_1] = [0, \boldsymbol{v} - 2\hat{\boldsymbol{n}}_1(\hat{\boldsymbol{n}}_1 \cdot \boldsymbol{v})]$$

Här användes vektoridentiteten $(\boldsymbol{a} \times \boldsymbol{b}) \times \boldsymbol{c} = \boldsymbol{b}(\boldsymbol{a} \cdot \boldsymbol{c}) - \boldsymbol{a}(\boldsymbol{b} \cdot \boldsymbol{c})$. Enligt sats 4 är produkten ovan en reflektion genom ett plan (innehållandes origo) med normalvektorn $\hat{\boldsymbol{n}}_1$. Den önskade rotationen fås nu genom att reflektera $q_{v'} = q_1 q_v q_1$ i plan 2 (ty, rotationsvinkeln är då θ), för den roterade kvaternionen q_r gäller därmed

$$q_r = q_2 q_{v'} q_2 = q_2 q_1 q_v q_1 q_2$$

Det återstår att beräkna $q = q_2 q_1$.

$$q = q_2 q_1 = [0, \hat{\boldsymbol{n}}_2][0, \hat{\boldsymbol{n}}_1] = [-\hat{\boldsymbol{n}}_2 \cdot \hat{\boldsymbol{n}}_1, \hat{\boldsymbol{n}}_2 \times \hat{\boldsymbol{n}}_1] = [-\cos\frac{\theta}{2}, -\sin\frac{\theta}{2}\hat{\boldsymbol{p}}] = [\cos\frac{\theta}{2}, \sin\frac{\theta}{2}\hat{\boldsymbol{p}}]$$

Notera att enligt sats 1 gäller $q^{-1} = (q_2 q_1)^{-1} = q_1^{-1} q_2^{-1} = \bar{q}_1 \bar{q}_2 = q_1 q_2$. □

Med sats 5 är det möjligt att arbeta med rotationer i kvaternionrummet och vi ser nu också fog för de variabelbyten som användes i föregående avsnitt 5.3.1.

5.3.3 Rotation med Eulervinklar

Eulers vinklar α, β och γ motsvarar rotationsvinklarna kring tre inbördes ortogonala axlar och därmed fås den resulterande matrisrepresentationen genom multiplikation av de elementära rotationsmatriserna (8). Men matrismultiplikation (och därmed rotation) är inte kommutativ vilket innebär att orienteringen givet (α, β, γ) ej är entydigt bestämd. Problemet löses genom att förelägga en rotationshierarki som bestämmer rotationsordningen, men tre möjliga axlar resulterar i 27 möjliga rotationshierarkier (XYZ, ZZY etc). Endast tolv av dessa kan representera en godtycklig rotation, ty frihetsgrader förloras då successiva rotationer sker kring samma axel [6]. Nedan betraktas hierarkin XYZ.

$$R_{xyz} = R(\alpha, \hat{x})R(\beta, \hat{y})R(\gamma, \hat{z}) \Rightarrow$$

$$R_{xyz} = \begin{bmatrix} \cos\beta\cos\gamma & -\cos\beta\sin\gamma & \sin\beta \\ \sin\alpha\sin\beta\cos\gamma & \cos\alpha\cos\gamma - \sin\alpha\sin\beta\sin\gamma & -\sin\alpha\cos\beta \\ -\cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma & \sin\alpha\cos\gamma + \cos\alpha\sin\beta\sin\gamma & \cos\alpha\cos\beta \end{bmatrix}$$

Låt nu $\beta = \pm\frac{\pi}{2}$ vilket leder till

$$R_{xyz} = \begin{bmatrix} 0 & 0 & \pm 1 \\ \frac{1}{2}(\sin(\alpha \pm \gamma) \pm \sin(\alpha \mp \gamma)) & \cos(\alpha \pm \gamma) & 0 \\ \mp\cos(\alpha \pm \gamma) & \sin(\alpha \pm \gamma) & 0 \end{bmatrix} = \mathbf{R}_{xyz}(\alpha \pm \gamma)$$

Ändring av α eller γ leder till rotation kring samma axel och därmed är en frihetsgrad förlorad. Ett system som hamnar i sådan position förlorar därmed tillgång till en mängd orienteringar och det är detta fenomen som kallas Gimbal-lock. Fenomenet är en inneboende singularitet i Eulerframställningen vilket innebär att alla rotationshierarkier har detta problem [6]. Eulers vinklar är ändå vanliga inom tillämpningar och möjliga anledningar till detta utreds kortfattat i nästa avsnitt.

5.3.4 Matris- eller kvaternionform?

Vi har nu redogjort för att orientering och rotation kan hanteras med matris- eller kvaternionoperationer. Det har också anmärkts att matrisrepresentationer med axel-vinkel och Eulervinklar ofta används inom tillämpningar vilket kan tyckas vara märkligt eftersom kvaternionrepresentationen kräver fyra tal och matrisrepresentationen nio. Eulerframställningen har dessutom en singularitet och en rotationshierarki att ta hänsyn till. Här följer en kort jämförelse mellan dessa metoder.

I texten Visualizing Quaternions analyseras beräkningskomplexiteten för fem operationer, matris till kvaternion, kvaternion till matris, rotera en 3-vektor, rotera flera 3-vektorer och sammansatta rotationer. Författaren visar att kvaternionoperationer endast är fördelaktigt vid det sistnämnda fallet men att kvaternioner är nödvändiga vid beräkning av optimal interpolering mellan ramar (p.g.a entydigt distansmått, kvaternionkurvor). Av numerisk osäkerhet och stabilitet kan det också vara en fördel att använda sig av kvaternionoperationer eftersom matrisrepresentationen innehåller överflödig information och fler villkor beaktas då en rotationsmatris måste ortogonaliseras och normaliseras medan en kvaternionoperator enbart behöver normaliseras [2] [3] [5].

Ovanstående gäller allmänt för matris- kontra kvaternionoperationer så låt oss närmare granska Eulerframställningen. Till dess fördel är en lång historia av utbrett användande och därmed finns utvecklade och testade lösningar vilket förenklar implementering. I allmänhet lämpar sig också matrisrepresentationer bättre för andra transformationer som t.ex. skjuvning [5]. För vissa system, som t.ex. gyroskop är det dessutom naturligt att använda Eulervinklar. Nackdelen är att ytterligare två villkor måste beaktas, Gimbal-lock och rotationshierarkin. Det är även svårt, givet tre vinklar och en hierarki, att förutsäga den resulterande rotationen (se kapitel 6.2). Detta visar sig matematiskt genom att det är relativt komplicerat att beräkna rotationsaxeln och -vinkeln. Detsamma kan sägas om att beräkna Eulervinklarna givet en rotationsmatris [5].

Kvatenionframställningen har länge varit känd men ej använts lika utbrett och därför kan lösningar och implementeringar saknas till många problem eller så används de inte på grund av bristande kunskap vad gäller färdighet eller existens. Utöver redan nämnda fördelar är att kvaternionen ger en kompakt framställning vilket förenklar konstruktion och avläsning av rotation [2] [5].

5.4 Visualisering

Kvaternioner är fyrdimensionella objekt vilket försvårar visualisering men under restriktion (6) kan problemet lösas med projektion till tre dimensioner. Här betraktas och jämförs egenskaperna för två olika projektionsmetoder, stereografisk projektion och kvadratrotsmetoden.

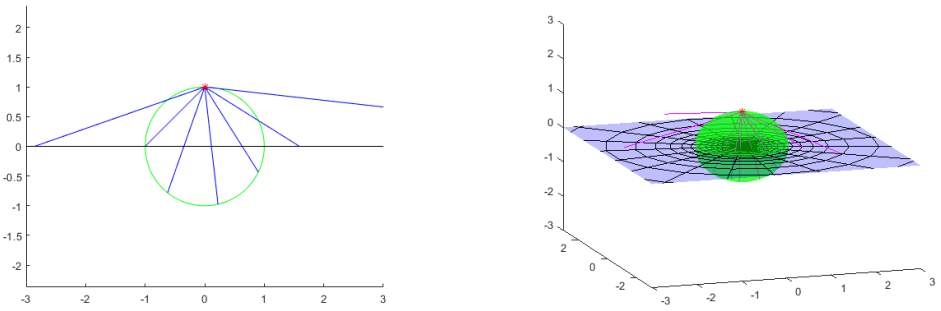
5.4.1 Stereografisk projektion

För att visualisera kvaternioner i 3D så behöver metoden ta bort en dimension vilket kan göras med stereografisk projektion. Innan metoden appliceras på enhetskvaternioner som ligger på hypersfären $\mathbf{S}^3$ så studeras projektionen av cirkeln $\mathbf{S}^1$ och sfären $\mathbf{S}^2$ på linjen respektive planet.

Den generella strategin för stereografisk projektion av ett objekt $\mathbf{O}$ går till på följande vis:

- Välj en projektionspunkt $\boldsymbol{p_0} \in \mathbf{O}$. I figur 4a och 4b valdes cirkelns och sfärens nordpoler.
- Välj ett projektionsplan π. I figur 4a och 4b valdes linjen $y = 0$ respektive planet $z = 0$.
- Skärningspunkten mellan planet π och linjen $\boldsymbol{l} = \boldsymbol{p_0} + t(\boldsymbol{p} - \boldsymbol{p_0})$ ger den stereografiska projektionen av $\boldsymbol{p} \in \mathbf{O}$ på π. I figur 4a och 4b är några av dessa linjer blå- respektive rödmarkerade.
- Projektionen av $\boldsymbol{p_0}$ definieras som *punkten i oändligheten* eftersom den annars saknar projektion.

Genom att studera figurerna 4a och 4b inses att det finns tre områden av intresse. Vid projektion av enhetscirkeln ser vi att norra halvcirkeln ($y > 0$) avbildas på $|x| > 1$ och södra halvcirkeln ($y < 0$) på $|x| < 1$. Till sist så har vi punkterna på ekvatorn ($y = 0$) som avbildas på sig själva, $|x| = 1$. Studium av projektionen av enhetssfären visar på samma vis att norra halvklotet ($z > 0$), det södra halvklotet ($z < 0$) och ekvatorn ($z = 0$) avbildas på områdena $x^2 + y^2 > 1$, $x^2 + y^2 < 1$ respektive enhetscirkeln $x^2 + y^2 = 1$.



(a) (b)

Figur 4: (a) Stereografisk projektion av enhetscirkeln på x-axeln. Figur: (b) Stereografisk projektion av enhetssfären på xy-planet.

Nu kommer vi till att projicera enhetskvaternionerna $q \cdot q = 1$ på det tredimensionella rummet av rena kvaternioner. Till skillnad från tidigare fall går det inte att direkt visualisera det projicerade objektet, men begränsningen (6) tillåter utläsning av alla komponenter. Likt de föregående fallen fås tre områden av intresse. Den norra halvan av ($q_0 > 0$) hypersfären avbildas i klotet ($q_1^2 + q_2^2 + q_3^2 < 1$), den södra halvan ($q_0 < 0$) utanför klotet ($q_1^2 + q_2^2 + q_3^2 < 1$), och ekvatorn ($q_0 = 0$) avbildas på enhetssfären ($q_1^2 + q_2^2 + q_3^2 = 1$).

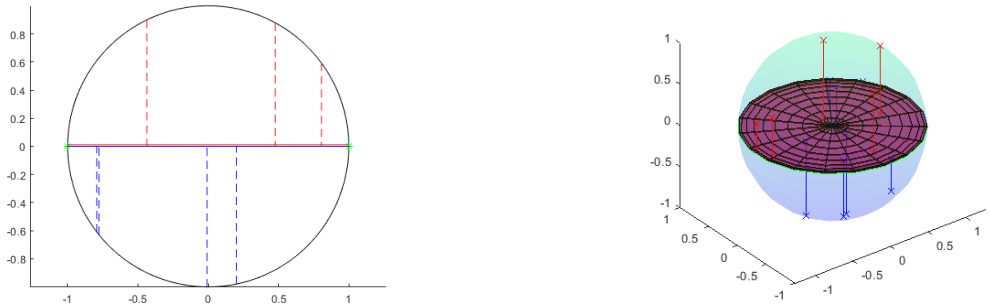
Fördelen med metoden är förståelse för objektets struktur och hur kurvor på hypersfären rör sig från ena halvan till den andra. Nackdelen är förvrängda proportioner och former [2] [7].

5.4.2 Kvadratrotsmetoden

Kvadratrotsmetoden utnyttjar att ytan $f(x_1, x_2, \ldots, x_n) = 0$ kan skrivas om enligt $x_n = g_i(x_1, \ldots, x_n - 1)$ där de olika g_i beskriver delytor. Givet en n-dimensionell yta behöver vi därmed specificera n-1 parametrar samt vilken av funktionerna g_i som används för att bestämma den n:te parametern. Vilken delyta som visualiseras kan i en grafisk representation till exempel färgkodas.

Hypersfären given $q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1$ skrivs nu som $q_0 = \pm\sqrt{1 - (q_1^2 + q_2^2 + q_3^2)}$. Låt g_1, g_2 och g_3 svara mot områden där $q_0 > 0$, $q_0 < 0$ respektive $q_0 = 0$. Detta resulterar i två enhetsklot vars ytor ($q_0 = 0$) överlappar. Vid denna omskrivning resulterar projektion av hypersfären i att enhetskvaternionens vektordel visualiseras som en punkt i enhetsklotet, men för att kunna utläsa fullständig information krävs även en indikation på tecknet för q_0. I figur 5 nedan visas resultatet av denna metod applicerad på enhetscirkeln och enhetsfären.

Kapitlet avslutas nu med jämförelse mellan de två projektionsmetoderna samt hur de kan komplementera varandra. Fördelen med kvadratrotsmetoden är att den i stora drag bevarar formen och proportionerna av en kurva på ytan men det är något svårare att visualisera vad som händer om kurvan ett flertal gånger går mellan sfärens två halvor. Att visualisera sådana kurvor i närheten av ekvatorn gör den stereografiska projektionen bra, men man bör ha i åtanke att den förvränger utseendet av kurvor beroende på var på sfären de befinner sig, särskilt kurvor som kommer nära $\boldsymbol{p_0}$ blir mycket utsträckta och därmed svåra att rita ut på liten yta [2].



(a) (b)

Figur 5: a) Projektion av enhetscirkeln på x-axeln med kvadratrotsmetoden: för $x \neq \pm 1$ $g_1 = \sqrt{1-x^2}$, $g_2 = -\sqrt{1-x^2}$ annars $g_3(\pm 1) = 0$. Avbildningarna visas med rött, blått och grönt i respektive ordning. b) Projektion av enhetssfären på xy-planet med kvadratrotsmetoden: för $z \neq 0$, $g_1 = \sqrt{1-(x^2+y^2)}$, $g_2 = -\sqrt{1-(x^2+y^2)}$
annars $g_3 = x^2 + y^2$. Avbildningarna visas med rött, blått och grönt i respektive ordning.

6 Tillämpningar av Grundteorin

Än så länge har vi granskat några av kvaternionens grundegenskaper och i detta kapitel ska dessa appliceras för att förklara två fenomen, Bält- och Boll-Tricket. Dessa företeelser kan inte förklaras genom användning av varken axel-vinkel-framställningen eller Eulers vinklar eftersom de saknar kvaternionens förmåga att beskriva rotationssekvenser.

6.1 Bält-Tricket

Bält-tricket är ett vanligt exempel på hur rotationssekvenser kan beskrivas. Metoden för tricket lyder som följer:

- Ett bälte läggs ut på en yta eller i rummet där båda ändar av bältet hålls fast.
- Sedan roteras bältet ett helt varv (2π) i en riktning, där ena sidan på bältet hålls fast. Därefter får bältet manipuleras för att invertera rotationen utan att orienteringen på ändarna får skilja sig från deras ursprungliga position.
- Procedur ovan utförs därefter igen fast med en rotation på 4π, i samma riktning.

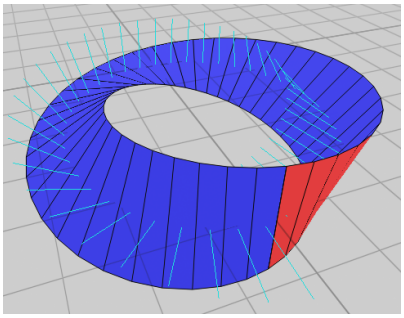
Det visar sig att med en 2π-rotation kan problemet inte lösas men för 4π är det möjligt att återgå till bältets startläge. Detta tar fram en viktig egenskap hos kvaternioner: spinorer [8]. Spinorer beter sig som vanliga vektorer i det komplexa rummet men när de roterar ett helt varv (2π) runt en kropp, kommer vektorn direkt att snurra runt tillbaka till sitt ursprungsläge, en viktig förmåga för att kunna beskriva t.ex. spin hos elektroner. Figur 6 illustrerar detta exempel. Bält-tricket kan förklaras både matematiskt och med visualisering. Låt bältet representeras som en mängd av anslutna punkter i kvaternionrummet $\mathbf{S}^3$. Låt också t parametrisera kurvan i $\mathbb{S}^3$ längs en av sidorna på bältet, där $t = 0$ och $t = 1$ blir ändpunkterna på bältet [2] sådan att

$$q(t) = q(w(t), x(t), y(t), z(t))$$

Vi kan rotera objektet i vardera riktning i 3D-rummet. Om vi låter y-axeln vara fixerad och roterar runt den med vinkeln θ längs axeln, fås att

$$q(t) = (\cos\frac{\theta t}{2}, 0, \sin\frac{\theta t}{2}, 0) \tag{14}$$

Matematiskt : Sätt in $\theta = 2\pi$ in i (14), vilket representerar ett helt varv av bältet runt y-axeln:



Figur 6: En representation av spinorer på ett Möbius-Band i Blender. Normalen till varje segmentyta visas av den blåa linjen.

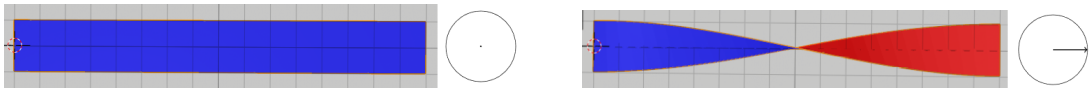
$$q(t) = (\cos \pi t, 0, \sin \pi t, 0) \Rightarrow \begin{cases} q(0) = (1, 0, 0, 0) \\ q(1) = (-1, 0, 0, 0) \end{cases} \tag{15}$$

Vi ser från ekvationen ovan att ett fullt varv av bältet inte har samma slutposition som den började med, vilket medför att vi ej kan återgå till begynnelsetillståndet. Notera att kvaternionramarna $q(0)$ och $q(1)$ motsvarar samma rotationsmatris vilket betyder att information förloras i matrisrepresentationen. Låt istället $\theta = 4\pi$ och sätt in i (14)

$$q(t) = (\cos 2\pi t, 0, \sin 2\pi t, 0) \Rightarrow \begin{cases} q(0) = (1, 0, 0, 0) \\ q(1) = (1, 0, 0, 0) \end{cases}$$

Därmed har vi en sluten kurva, där både start- och sluttillståndet befinner sig med samma orientation.

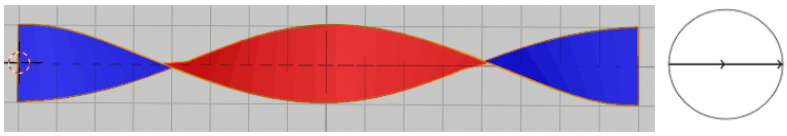
Visualisering : För visualiseringen av tricket kommer spinorer, som togs upp tidigare i kapitlet, till bra hjälp. Spinorer och deras rotationer representeras oftast på formen av diskar samt riktningslinjer. I detta fall sätts radien till π, vilket beskriver att varje rotation av multipeln π blir längden mellan centrum och omkretsen av disken. En simpel demonstration av detta är att lägga bältet på ett bord. Vi kan dela upp disken i två segment, ett som beskriver rotationen parallellt till bordet och ett annat parallellt till bältet. Eftersom vi enbart ska undersöka en av rotationerna i spelet, tas vridningen parallellt med bältet som referens. Spinorernas axlar sätts fast längs den roterande änden hos bältet eftersom orienteringen skall vara densamma här efter att en rotation har genomförts. Figur 7 nedan visar spinorens funktion.



(a) (b)

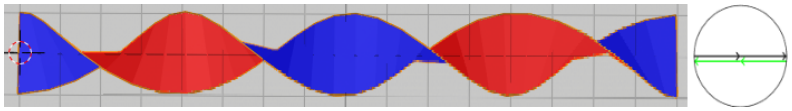
Figur 7: Visualisering av rotationer med spinorer. Figur a) visar bältet i startposition utan någon vridning. I figur b) har bältet roteras med π åt ett håll, där spinoren visar en pil i vågrätt led med längden π. Den röda cirkeln markerar axeln som inte roteras i varje figur.

Notera att en rotation i $-\pi$-riktning hade gett en pil mot vänster istället för höger från referenspunkten. Således ger en 2π-rotation två pilar som pekar i samma riktning, se figur 8 nedan. Givet en $\theta = 2\pi$-rotation kring y-axeln, ges följande tillstånd:



Figur 8: En 2π-rotation längs bältets längd med spinorens orientering.

I figur 8 avläses att spinoren inte befinner sig i mittcirkeln, vilket indikerar att bältet har fel orientering i rummet. Detta stämmer också överens med ekvation (15) i den matematiska delen som tyder på att vi inte har en sluten kurva. Däremot, som det nämndes innan, kommer spinorerna efter 2π att flippa runt, det vill säga att de kommer byta riktning från höger till vänster. Då $\theta = 4\pi$, har vi den följande representationen:



Figur 9: En 4π-rotation längs bältets längd med spinorens orientering.

Därmed fås samma orientering hos bältet efter en 4π-rotation som i figur 7 a), eftersom enligt spinorerna tar riktningar ut varandra och man kommer fram till startpositionen. Detta kan ses som att en rotation av 4π inte är en rotation alls och istället betraktas som en sluten kurva i kvaternionrummet $\mathbf{S}^3$.

6.2 Boll-Tricket

Ännu ett exempel då ordningen av en sekvens rotationer har betydelse är det så kallade boll-tricket. Det går till på följande vis:

- Placera en boll på ett bord och lägg handen på bollen parallellt med bordsytan. Låt bordsytan ligga i x,y-planet med z-axeln riktad uppåt, ut ur bordet.
- Rör nu handen i små cirklar medurs och notera vad som händer med bollen.

Bollen roterar långsamt moturs runt z-axeln som är ortogonal mot bordsytan trots att vi aldrig direkt roterar bollen runt den axeln. I figur 11 visualiseras projektionen av bollens orientering där dess begynneslseram svarar mot kvaternionen $q[1, \mathbf{0}]$. I avsnitt 5.3.1 konstaterades att kvaternionmultipliktation motsvarade ramrotation, kurvan i figuren visualiserar därmed en sekvens av kvaternionmultiplikationer. Rotation medurs i xy-planet resulterar i sekvensen $q[\cos\frac{\theta(t-dt)}{2}, \sin\frac{\theta(t-dt)}{2}\hat{\boldsymbol{\theta}}(t-dt)]...q[\cos\frac{\theta dt}{2}, \sin\frac{\theta dt}{2}\hat{\boldsymbol{\theta}}(dt)]q[1, \mathbf{0}]$ där $\hat{\boldsymbol{\theta}}(t) = \sin\theta t\hat{\boldsymbol{x}} - \cos\theta t\hat{\boldsymbol{y}}$ och θ en konstant liten rotationsvinkel. Detta märks ännu tydligare ifall bollen först roteras $\frac{\pi}{2}$ runt negativa y-axeln, följt av $\frac{\pi}{2}$ runt x-axeln och slutligen $\frac{\pi}{2}$ runt y-axeln. Resultatet av den sekvens kan ses i figur 10.

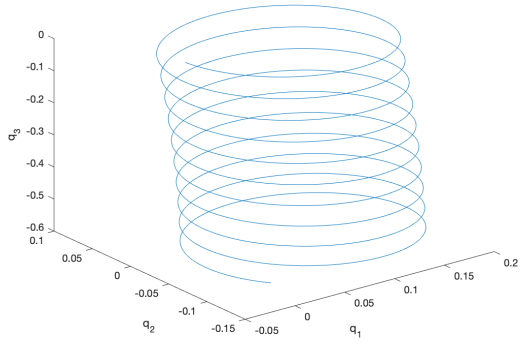
(a) (b) (c) (d)

Figur 10: Visualisering av sekvensen av rotationer som refereras i exempel 2, färg och skiva indikerar orientering av bollen.

Detta sker eftersom rotationer ej är kommutativa, annars hade den resulterande rotationen av det andra exemplet varit $\pi/2$ runt x-axeln och kurvan i figur 11 hade varit sluten i q_1q_2-planet. Rotationer kring x- och y-axlarna kan ge upphov till rotation kring z-axeln vilket också kan inses med tillämpning av Eulers vinklar med rotationshierarkin YXY, se avsnitt 5.3.3 [2].

7 Avancerad Teori

I detta kapitel undersöks mer avancerade egenskaper hos kvaternionen och hur dessa kan visualiseras. Speciellt utforskas kvaternionens relation till 3D-ytor.



Figur 11: Kvaternionkurvan för långsam cirkulär rörelse med projektionen av kvaternionramens imaginärdel. Lägg märke till att kurvan ej är sluten och sträcker sig i q_3-led vilket betyder att bollens orientering roterats i z-led.

7.1 3D-ytor

I detta avsnitt utreds hur 3D-ytor kan beskrivas med kvaternioner. Vi kommer använda oss av kvaternionramar (se avsnitt 5.2) för att beskriva 3D-ytorna. Kvaternionramar ger oss möjligheten att beskriva alla typer av ytor och samtidigt karaktärisera ytans textur, vilket generar diverse användningsområden. Mer precist kommer vi i denna rapport endast behandla släta differentierbara 3D-ytor. Teorin som presenteras nedan kan även justeras för att utvidgas till fyra dimensioner [2].

Vi använder oss av rotationsmatrisen R_q (10) och rena kvaternioner för att få de exakta komponenterna av R_q, vilket är $qv\bar{q} = q(0, \mathbf{v})\bar{q} = (0, R_q\mathbf{v})$. Vi använder sambandet mellan 3×3 matrisen R_j^i och en godtycklig kvaternion q enligt:

$$R_q(\mathbf{V})^i = \sum_j R_j^i V^j = q(0, V^i)q^{-1}.$$

Varje ortonormal ramkomponent kan beskrivas som en kolumn av R_j^i och en godtycklig ram fås då genom att applicera sats 5 på de tre kartesiska axlarna (representerade med rena kvaternioner):

$$\begin{aligned} \hat{\mathbf{T}}_1 &= q(0, \hat{\mathbf{x}})q^{-1}, \\ \hat{\mathbf{T}}_2 &= q(0, \hat{\mathbf{y}})q^{-1}, \\ \hat{\mathbf{N}} &= q(0, \hat{\mathbf{z}})q^{-1}. \end{aligned} \tag{16}$$

Vi skriver det som en matris av kvaternioner i kvadratisk form där ramvektorerna är kolumner.

$$\left[[\hat{\mathbf{T}}_1] \quad [\hat{\mathbf{T}}_2] \quad [\hat{\mathbf{N}}] \right] = R_q. \tag{17}$$

Genom att ta differentialen och använda oss av hastighets normaliseringen $v(t) = \|\mathbf{x}'(t)\|$ så kan kvaterniondifferentialen skrivas som:

$$\begin{bmatrix} q_0' \\ q_1' \\ q_2' \\ q_3' \end{bmatrix} = v(t)\frac{1}{2} \begin{bmatrix} 0 & -k_x & -k_y & -k_z \\ k_x & 0 & k_z & -k_y \\ k_y & -k_z & 0 & k_x \\ k_z & k_y & -k_x & 0 \end{bmatrix} \cdot \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}. \tag{18}$$

Om $\mathbf{k} = 2(q_0\mathbf{dq} - \mathbf{q}dq_0 - \mathbf{q} \times \mathbf{dq})$ får vi:

$$\begin{aligned} d\hat{\mathbf{T}}_1 &= q(0, \mathbf{k} \times \hat{\mathbf{x}})q^{-1}, \\ d\hat{\mathbf{T}}_2 &= q(0, \mathbf{k} \times \hat{\mathbf{y}})q^{-1}, \\ d\hat{\mathbf{N}} &= q(0, \mathbf{k} \times \hat{\mathbf{z}})q^{-1}. \end{aligned} \tag{19}$$

Nu har vi formulerat ramekvationer med kvaternioner och dessa kan användas för att beskriva egenskaper hos 3D-ytor. Först skall vi studera en parameterrepresentation av ytor på klassiskt vis:

$$\left[\frac{\partial \hat{\mathbf{N}}(u,v)}{\partial u} \quad \frac{\partial \hat{\mathbf{N}}(u,v)}{\partial v}\right] = \left[\hat{\mathbf{T}}_1(u,v) \quad \hat{\mathbf{T}}_2(u,v)\right][\kappa], \tag{20}$$

Egenvärdena till matrisen $[\kappa]$ är ytans huvudkrökningar k_1 och k_2, det vill säga den maximala och minimala krökningen i en given punkt. Då är $K = det\,[\kappa] = k_1 k_2$ vår Gauss-krökning och $H = \frac{1}{2} tr\,[\kappa] = \frac{k_1+k_2}{2}$ är den genomsnittliga krökningen [2].

För att beskriva samma ytegenskaper som ovan men med kvaternioner så används uttrycket för differentialen (18) och ramparametriseringen $(\mathbf{a}, \mathbf{b})$:

$$\begin{aligned} q_u &\equiv \tfrac{\partial q}{\partial u} = \tfrac{1}{2} q(0, \mathbf{a}), \\ q_v &\equiv \tfrac{\partial q}{\partial v} = \tfrac{1}{2} q(0, \mathbf{b}). \end{aligned}$$

Vi kan nu skriva om ekvation (20) med kvaternioner för att beskriva krökningen, vi använder standardnotation från differentialgeometrin där K och H är lokala krökningar. Därmed kan de lokala krökningarna skrivas i termer av ett godtyckligt oberoende linjärt par av vektorfälten $(\mathbf{U}, \mathbf{V})$ som

$$D_{\mathbf{U}}\hat{\mathbf{N}} \times D_{\mathbf{V}}\hat{\mathbf{N}} = K(\mathbf{U} \times \mathbf{V}), \tag{21}$$

$$D_{\mathbf{U}}\hat{\mathbf{N}} \times \mathbf{V} + \mathbf{U} \times D_{\mathbf{V}}\hat{\mathbf{N}} = 2H(\mathbf{U} \times \mathbf{V}), \tag{22}$$

där $\mathbf{U} = \mathbf{x}_u \cdot \bigtriangledown$ och $\mathbf{V} = \mathbf{x}_v \cdot \bigtriangledown$. Genom att använda ekvationerna (21), (22), kvaternionidentiteten $\hat{\mathbf{I}} = (1, \mathbf{0}) = q(1, \mathbf{0})q^{-1}$ och (16) får vi:

$$q_u q_v^{-1} = -\frac{1}{4} q(0, \mathbf{a})(0, \mathbf{b}) q^{-1} = -\frac{1}{4} q(-\mathbf{a} \cdot \mathbf{b}, \mathbf{a} \times \mathbf{b}) q^{-1}$$

$$= -\frac{1}{4}[-\mathbf{a} \cdot \mathbf{b}\hat{\mathbf{I}} + (\mathbf{a} \times \mathbf{b})_x \hat{\mathbf{T}}_1 + (\mathbf{a} \times \mathbf{b})_y \hat{\mathbf{T}}_2 + (\mathbf{a} \times \mathbf{b})_z \hat{\mathbf{N}}_1. \tag{23}$$

koefficienten framför $\hat{\mathbf{N}}$ är vår Gauss-krökning $(\mathbf{a} \times \mathbf{b})_z = K$. Den genomsnittliga krökningen ges på likande vis av:

$$q(0, \hat{\mathbf{x}}) q_u^{-1} + q(0, \hat{\mathbf{y}}) q_v^{-1} = -\frac{1}{2} q(-\hat{\mathbf{x}} \cdot \mathbf{a} - \hat{\mathbf{y}} \cdot b, \hat{\mathbf{x}} \times \mathbf{a} + \hat{\mathbf{y}} \times b)$$

$$= -\frac{1}{2}\left[-(a_x + b_y)\hat{\mathbf{I}} + b_z \hat{\mathbf{T}}_1 - a_z \hat{\mathbf{T}}_2 + (a_y - b_x)\hat{\mathbf{N}}\right]. \tag{24}$$

där $(a_y - b_x) = tr[\kappa] = 2H$ är uttrycket för vår genomsnittliga krökning. Projektionen mot normalens $\hat{\mathbf{N}}$ riktning av våra ekvationer (23) och (24) ger oss vår Gauss- och genomsnittliga krökning:

$$K = det \begin{bmatrix} a_y(u,v) & -a_x(u,v) \\ b_y(u,v) & -b_x(u,v) \end{bmatrix} = a_x b_y - a_y b_x,$$

$$H = \frac{1}{2} tr \begin{bmatrix} a_y(u,v) & -a_x(u,v) \\ b_y(u,v) & -b_x(u,v) \end{bmatrix} = \frac{1}{2}(a_y - b_x).$$

Vi kan även utöka liknande ekvationer till 4D genom vår differens (18) [2].

7.2 Visualisering av 3D-ytor

Vi vill nu visualisera teorin från avsnitt 7.1. Vi kommer använda oss av Gaussavbildning för att avbilda kvaternionramar. Gaussavbildning är i stora drag en nätmaskindelning av en enhetssfär där varje knutpunkt är normalens riktning av motsvarande nod på en indelad yta. Ramar behöver inte användas för att genomföra en Gaussavbildning utan det skulle vara tillräckligt att använda enbart ytans noder och normaler, men då försvinner en del egenskaper. En central

egenskap är möjligheten att applicera egenskaper på ytan, exempelvis en textur. Dessutom kan man alltid vid givet koordinatsystem konvertera systemet till kvaternionramar och därmed erhålla egenskaperna hos kvaternioner. Eftersom den klassiska beskrivningen av ramar och projektionen av koordinaten befinner sig i samma rum som kvaternionramarna, båda planen är nämligen vinkelrätta med $\hat{\mathbf{z}}-$axeln [2].

Innan kvaternioner tillämpas för Gaussavbildning ges först den klassiska definitionen av Gaussavbildning:

Definition 7. *Låt $S \subset \mathcal{R}^3$ vara en yta med en orientering N. Avbildningen $N : S \to \mathcal{R}^3$ tar sina värden i enhet 2-sfären S^2. Avbildningen $N : S \to S^2$ kallas* ***Gaussavbildning*** *av ytan S [9].*

Vi kan använda oss av att Gaussavbildningen är differentierbar i punkten $p \in S$ och att $dN_p : T_pS \to T_{N(p)}S^2$ är en linjär avbildning för att få informationen om krökningen. Vi vet att vi då befinner oss i ett delrum av R^3 och att $T_p = t_{N(p)}S^2$. Därmed kan vi skriva $dN_p : T_pS \to T_pS$. Informationen om krökningen av en yta ges då av diffrentieringen av normalen som i (20) [9].

Vi ska nu beskriva hur man kan använda Gaussavbildning med kvaternioner. Det är dock en komplicerad process att differentiera R_q som vi gjorde med den klassiskt beskrivna ramen eftersom R_q är kvadratisk och samtidigt består av alla kvaternionkomponenter. Istället används en projektion till ett delrum av kvaternionrummet baserat på bilinjäriteten av kvaternioner på rena vektorer genom att använda oss av exempelvis Hopf avbildning, vilket är ett sätt att beskriva $\mathbf{S}^3$ med hjälp av cirklar och en vanlig sfär. Vi kan alltså använda oss av delar för att konstruera hela vår avbildning. Dessa delar är så kallad lappytor och ur klassisk avbildningsteori fås att det behövs åtminstone två separata lappytor för att kunna placera en komplett mängd av koordinater. För kvaternion-Gaussavbildning så använder man kvaternionramar istället för koordinater, därmed gäller att för varje punkt $\mathbf{x}$ av en lappyta så har vi en avbildning som går från vår lappyta till $\mathbf{S}^3$. Relationen mellan två kvaternionramar q och q' av två närliggande lappytor U och U' ges av kvaternionmultiplikation med en övergångsfunktion t:

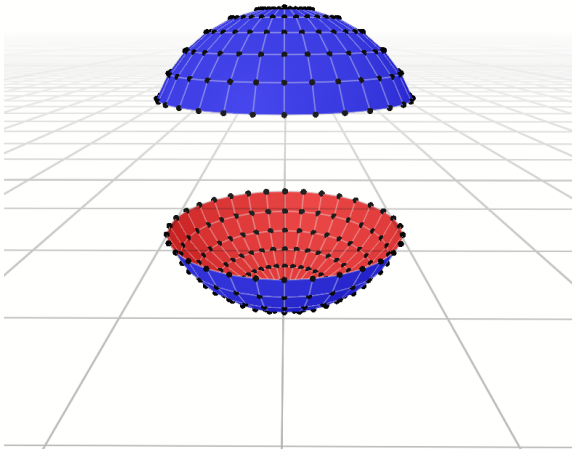
$$q' = tq.$$

Detta ger oss en explicit övergångsfunktion mellan två lappytor. Det är ett liknande tillvägagångssätt för att beskriva en komplett avbildning, men då måste man också ta hänsyn till den explicita relationen mellan kvaternionramar för varje lappyta som de delar i varje punkt. En sådan relation kan beskrivas enligt:

$$t(\theta) = q_{ytlapp_1}(\theta)q^{-1}_{ytlapp_2}(\theta)$$

där varje punkt som lappytorna delar med varandra, d.v.s. $U \cap U'$ är parametriserad av θ.

Det finns olika optimeringsstrategier för hur man ska välja ramar och i sin tur lappytor för att exempelvis ta bort onödiga vridningar och vändningar av formen. Intuitivt kan man tänka sig att man vill minimera antalet vridningar som ramkurvan behöver ta sig från startpunkten på ramen till slutpunkten på ramen. Denna strategi kallas för *Minimal längd och area strategin* och visualiseras i figur 12.



Figur 12: Exempel: En sfär uppdelad i två ytlappar, en sydpol och en nordpol, med Minimal längd och area strategin.

8 Tillämpning av Avancerad Teori

I detta kapitel tillämpas kvaternionteorin i en kollisionssimulering. För att kunna använda teorin ovan så begränsar vi oss till kollisioner mellan konvexa former. Läsaren påminns om att vårt syfte inte är att producera en effektiv kollisionsalgoritm utan att visa hur kvaternioner och dess visualisering kan tillämpas.

8.1 Kollisiondetektion

Vi använder oss av Minkowski-differensen för att se om två konvexa former är i kollision. Vi börjar med att definiera Minkowski-differensen:

Definition 8. *Låt A och B vara två konvexa objekt. Minkowski-summa av A och B, $A, B \in R^3$:*

$$A \oplus B = \{a + b | a \in A, b \in B\}$$

Om vi sätter $-B = \{-b | b \in B\}$ så får vi Minkowski-differensen:

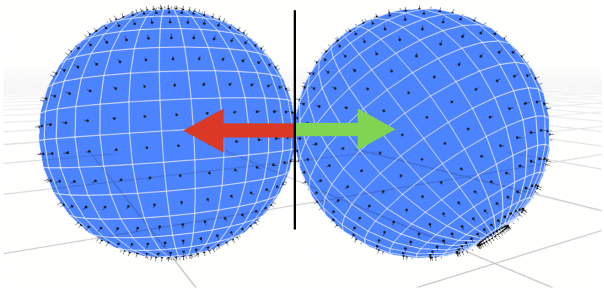
$$A \ominus B = A \oplus (-B) = \{a - b | a \in A, b \in B\}.$$

Vi uppmärksammar att Minkovski-differensen $B \ominus A$ också är en konvex form, N, vilket ger att om följande villkor är uppfyllt så har vi en kollision.

$$N_{B \ominus A}(\mathbf{n}) = N_B(\mathbf{n}) - N_A(-\mathbf{n}).$$

Vi ser då att om det sker en kollision så måste $N_{B \ominus A}(\mathbf{n})$ innehålla minst en punkt [10].

Anmärkning: Vi kan använda diverse metoder för att beskriva konvexa former, ett alternativ är Superquadratic, se appendix B.



Figur 13: Exempel: En visualisering av två Superquadratic objekt, vars Minkowski-differens innehåller minst en punkt. Dessutom är normalerna visualiserade.

8.2 Kollision med friktion

Det vore enklare att undersöka friktionsfri kollision till skillnad från med friktion, eftersom det då är möjligt att direkt lösa för kollisionskraften men vid makroskopiska kollisioner ger detta ej verklighetstrogna resultat [3] [11]. Å andra sidan är det svårt att beräkna kraften för kollision med friktion. En lösning som nedan redovisas ges av en metod utvecklad av Brian Mitrich som istället för kraft beräknar impulsen [11]. Det är dock de inblandade kvaternionerna som ska visualiseras, inför simulering är det därför lämpligt att beskriva en kropps dynamiska tillstånd i termer av kvaternioner.

8.2.1 Dynamiskt tillstånd och kvaternionform

Betrakta ett dynamiskt tillstånd $\boldsymbol{x}(t)$:

$$\boldsymbol{x}(t) = \begin{bmatrix} \boldsymbol{r}_{cm}(t) \\ R(t) \\ \boldsymbol{v}(t) \\ \boldsymbol{\omega}(t) \end{bmatrix} \tag{25}$$

där $\boldsymbol{r}_{cm}$ är masscentrumposition, R orientering, $\boldsymbol{v}(t)$ hastighet och $\boldsymbol{\omega}$ vinkelhastighet. Tillståndet innehåller alltså all nödvändig information för kollisionsdetektion och kollisionssimulering. För att studera tillståndets beteende i kvaternionrummet översätts alla variabler till kvaternionform. Alltså, vektorerna representeras av rena kvaternioner och orienteringen (rotationsmatrisen) av korresponderande enhetskvaternion:

$$\boldsymbol{x}_q(t) = \begin{bmatrix} q_{cm}(t) \\ q(t) \\ q_v(t) \\ q_\omega(t) \end{bmatrix} \tag{26}$$

där $q_{cm}(t) = [0, \boldsymbol{r}_{cm}(t)]$, $q_v(t) = [0, \boldsymbol{v}(t)]$, $q_\omega(t) = [0, \boldsymbol{v}(t)]$ och $R(t) = R(q(t)) \rightarrow q(t)$. Om tillståndets komponenter normaliseras så kan de visualiseras (se kapitel 5.4).

8.2.2 Impulsbaserad kollisisonsmodell

Modellen presenteras genom en sats vars bevis läsaren finner i appendix eller i artiklarna [11] [13]. Modellen baseras på tre antaganden vilka först presenteras och därefter följer satsen.

Antagande 1: *Infinitesimal kollisionstid*
Antagandet tillåter att kropparnas rörelse behandlas som konstant under kollisionsförloppet. Rimligt eftersom kroppsrörelsen ofta är försumbar under kollisionstiden.

Antagande 2: *Poissons hypotes*

Kontaktytan upplever vid kollision en kompression- och en expansionsfas, den totala impulsen p antas vara relaterad till kompressionsimplulsen p_k enligt

$$p = (1 + \epsilon)p_k \tag{27}$$

där ϵ är restitutionkoefficienten. Modellen kommer från empiriska test [13].

Antagande 3: *Coulombfriktion*
För en friktionskoefficent μ, en friktionskraft $\boldsymbol{f_f}$ och kollisionpunktens tangethastighet $\boldsymbol{u}_t$ antas det vid glidning gälla

$$\boldsymbol{u}_t \neq 0 \Rightarrow \boldsymbol{f}_f = -\mu \boldsymbol{f}_n \hat{\boldsymbol{u}}_t \tag{28}$$

och vid rullning

$$\boldsymbol{u}_t = 0 \Rightarrow |\boldsymbol{f}_f| \leq -\mu |\boldsymbol{f}_n|. \tag{29}$$

Dessa relationer är härledda från experiment.

Sats 6. *Vid en stelkroppskollision under antaganden 1,2 och 3 fås förändringen av en kropps dynamiska tillstånd genom att integrera av ekvation* (30) *vid glidning och ekvation* (31) *vid rullning[13].*

$$\Delta \boldsymbol{u}' = \begin{bmatrix} u_x'(p_z) \\ u_y'(p_z) \\ u_z'(p_z) \end{bmatrix} = M \begin{bmatrix} -\mu \frac{u_x(p_Z)}{\sqrt{u_x^2(p_Z)+u_y^2(p_Z)}} \\ -\mu \frac{u_y(p_Z)}{\sqrt{u_x^2(p_Z)+u_y^2(p_Z)}} \\ 1 \end{bmatrix} = M\boldsymbol{p}'. \tag{30}$$

$$\begin{bmatrix} 0 \\ 0 \\ u_z' \end{bmatrix} = M \begin{bmatrix} \alpha \\ \beta \\ 1 \end{bmatrix} \tag{31}$$

där

$$\Delta \boldsymbol{v}(p_z) = \frac{1}{m} \boldsymbol{p}(p_z) \tag{32}$$

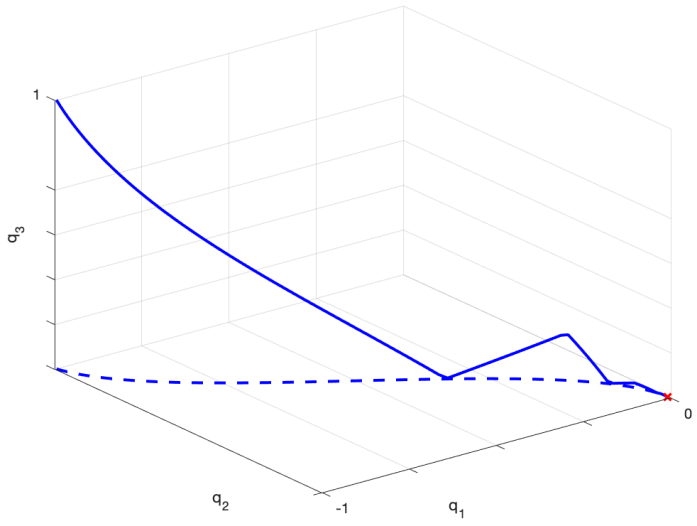
$$\Delta \boldsymbol{\omega}(p_z) = \boldsymbol{J}^{-1}(\boldsymbol{r} \times \boldsymbol{p}(p_z)). \tag{33}$$

Där $\alpha^2 + \beta^2 \leq \mu^2$ och integrationen görs m.a.p på impulsen p_z, koordinatsystemet är valt så att z-riktningen är parallell med normalriktningen vid kollisionspunkten och matrisen M beror på massan och massfördelningen hos kropparna.

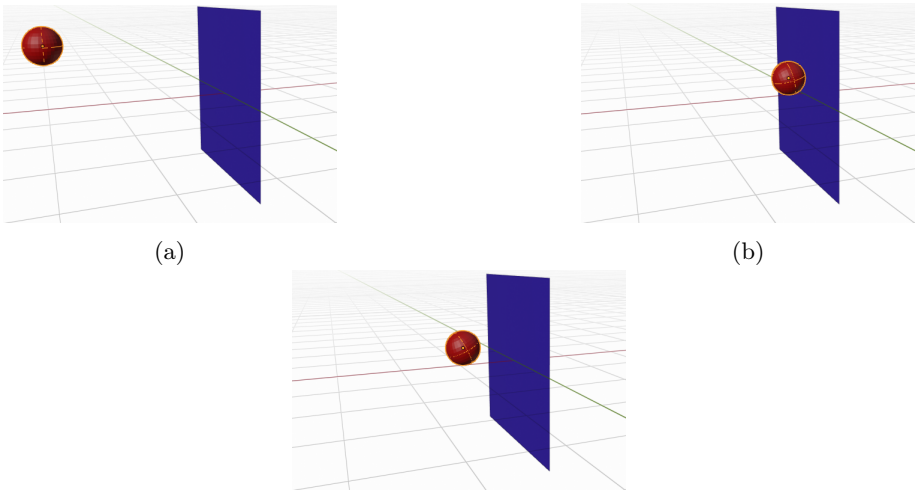
För beräkning av matrisen M hänvisas läsaren återigen till appendix eller till artiklarna [11] [13]. Kollsisionsdetektionen beskriven i föregående kapitel används även till att finna kollisionssystemet, d.v.s kvaternionen vilka xyz-riktningar som korresponderar mot p_z, u_x, u_y och u_z. Därefter integreras ekvationerna ovan numeriskt tills $u_z = 0$, då är kompressionsfasen över och enligt antagande gäller $p_{zf} = (1 + \epsilon)p_z$, integrationen fortsätter alltså tills $p_z = p_{zf}$ och vi får till slut den totala impulsen $\boldsymbol{p}(p_{zf})$(från ekvation (30) och (31)) vilken används i ekvationerna (35) och (36) för finna kropparnas nya hastighet och vinkelhastighet.

8.3 Visualisering av en kollision

I detta avsnitt visualiseras två kollisioner genom simulering med hjälp av teoribeskrivningen från avsnitt 8.2. En kollision mellan två sfärer med samma egenskaper men olika begynnelsetillstånd. En andra simulering gjordes för en sfär som kolliderar mot ett plan (se figur 15). Kvaternionen svarande mot sfärens orientering ses i figur 14. En kortfattad beskrivning på hur vi genomfört vardera simulering samt kod kan avläsas i Appendix C respektive D.1.5 och D.1.6.



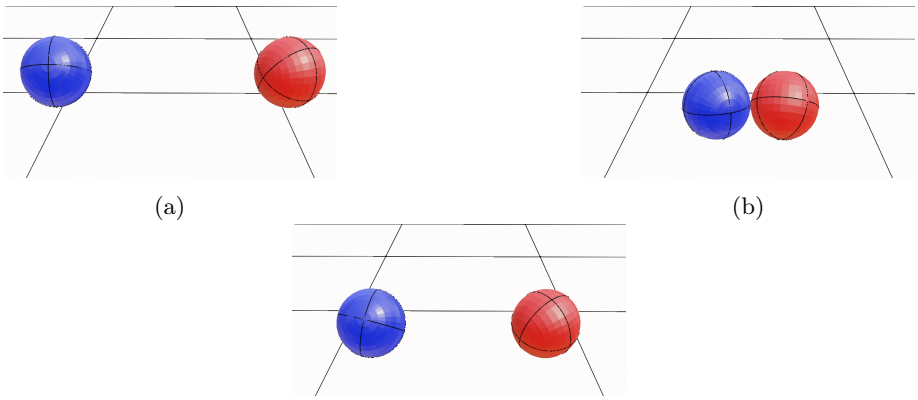
Figur 14: Orienteringen hos sfären med kollision mot ett plan. Den blåa linjen representerar banan innan kollisionen (röda punkten) och den streckade är vad som sker efter kollisionen med planet. Notera att kurvans riktningsförändring innan och efter kollisionen beror på kollision med golvet.



(a) (b)

(c)

Figur 15: Simulering av en kollision mellan ett plan och en sfär. Figur a) visar startpositionerna medan i b) sker kollisionen. Figur c) visar vad som sker efter kollisionen har genomförts.



(a) (b)

(c)

Figur 16: Simulering av en kollision mellan två sfärer. Figur a) och b) visar objekten precis innan och under kollisionsmomentet. I figur c) har kollisionen utförts.

9 Slutsatser och Diskussion

Resultatet av denna litteraturstudie visar att kvaternioner kan användas för att beskriva rotation och orientering samt att de kan vara mer lämpliga att använda än Eulervinklar. En anledning till detta är Eulervinklarnas inneboende singularitet kallad Gimbal-lock, som kvaternionframställningen inte delar. Det finns dock tillämpningar för att använda Eulers vinklar då rörelsen är begränsad. Matrisoperationer är ofta mer effektiva vilket resulterar i att det kan vara fördelaktigt att växla mellan metoderna. Kvaternionen avbildar rotationsmatriser (eller ramar) till punkter vilket gör det möjligt att definiera ett entydigt likhetsmått (distansen i $\mathbf{S}^3$), och rotationssekvenser avbildas som kurvor. Dessa egenskaper tillåter oss att förstå fenomen Bält- och Boll-tricket men det finns mer kraftfulla tillämpningar av dessa egenskaper som vi inte utrett. Ett exempel gäller att finna en optimal eller differentierbar rotationssekvens mellan två ramar, vilket kan göras genom att studera och interpolera kvaternionkurvor. Detta är viktigt inom animation för att skapa mjuka rörelser.

För att visualisera kvaternioner studerades två typer av projektion till tre dimensioner, stereografisk projektion och projektion med kvadratrotsmetoden. Kvadratrotsmetoden bevarar formen och proportionerna bättre men kan samtidigt vara mer komplicerad att visualisera i jämförelse med stereografisk projektion.

Användning av kvaternioner inom differentialgeometri kan vara fördelaktigt för att beskriva komplexa ytor, då användandet av kvaternioner erhåller de fundamentala fördelarna med kvaternioner. Dessutom kan teorin utvidgas till fyra dimensioner på ett naturligt sätt, då kvaternioner i grunden är fyrdimensionella. Kvaternion-Gaussavbildning avbildar komplexa kvaternionramar genom användning av så kallade lappytor. Tillämpningen av kvaternioner inom differentialgeometrin fastställer fördelaktiga avseenden med kvaternioner, mer specifikt i sammanhang av rapportens syfte; visualisering och användning.

Syftet med denna rapport var att åskådliggöra tillämpning och visualiserng av kvaternioner vilket har gjorts men ingen ny kunskap har presenterats. Kvaternionen har en lång historia och är ett pågående forskningsfält och därmed är kunskapen förmedlad i denna rapport mycket begränsad.

Referenser

[UR] Utformning av rapporter och kandidatarbetens skriftliga presentation för Civilingenjörsprogrammen vid Chalmers tekniska högskola. 2008. Göteborg: Chalmers Tekniska Högskola.

[1] Coutsias E, Romero L. The Quaternions with an application to Rigid Body Dynamics [Internet]. New Mexico: Department of Mathematics and Statistics; 1999. Hämtad från: http://www.ams.stonybrook.edu/ coutsias/papers/rrr.pdf.

[2] Andrew J. Hanson. Visualizing Quaternions: Series in interactive 3D technology. San Francisco: Morgan Kaufmann Publishers; 2006.

[3] Wang H. Rigid Body Dynamics [Opublicerat material]. Hämtad från: $http://web.cse.ohio-state.edu/\ wang.3602/courses/cse3541-2017-fall/rigid_note.pdf$ (2017).

[4] Vince J. Quaternions for Computer Graphics. 1 ed. New York: Springer; 2011.

[5] Dam E, Koch M, Lillholm M. Quaternions, Interpolation and Animation. Köpenhamn: Deparment of Computer Science, University of Copenhagen; 1998.

[6] Sciliano B, Sciavicco L, Oriolo G. Robotics - Modelling, Planning and Control. 1 ed. London: Springer; 2010.

[7] Eater B, Sanderson G. Visualizing quaternions, An explorable video series;2018 [citerad 2020-05-14]. Hämtad från: https://eater.net/quaternions

[8] Mark Staley. Understanding Quaternions and the Dirac Belta Trick [Internet]. New York: Cornell University; 2010. Hämtad från: https://arxiv.org/abs/1001.1778?context=physics.

[9] Kamberova, George & Kamberova, Gerda. Recovering Surfaces from the Restoring Force [Internet]. New Jersey & New York: Stevens Insitue of technology & Hofstra univerisy; 2004. Hämtad från: $https://cs.hofstra.edu/courses/2004/spring/csc120/01/reappt/paper702_print_optimized.pdf$.

[10] Jacobs, Scott. Game Programming Gems 7. Course Technology; 2008.

[11] Mirtich B, Canny J. Impulse-based Simulation of Rigid Bodies [Internet]. California: University of California at Berkeley; 1995. Hämtad från: https://graphics.stanford.edu/courses/cs468-03-winter/Papers/ibsrb.pdf.

[12] Laura Downs. A Proof of the Quaternion Rotation Representation. Berkeley EECS. 2003 (2019-03-02).

[13] Mirtich B, Canny J. Impulse-based Dynamic Simulation [Internet]. California: University of California at Berkeley; 1995. Hämtad från: https://graphics.stanford.edu/courses/cs468-03-winter/Papers/ibds.pdf.

[14] Barr, Alan. Superquadrics and Angle-Preserving Transformations in IEEE Computer Graphics and Applications, vol. 1, no. 1, Jan. 1981, doi: 10.1109/MCG.1981.1673799.

Bilagor

A Bevis för sats 6

Bevis. Notation för att kunna särskilja de olika kropparna och deras tillstånd vid kollisionsögonblicket. Ett heltals index i används för att beteckna tillståndsvariabeler tillhörnandes t.ex. kropp 1.
$m_i =$ *Massan för kropp i.*
$\boldsymbol{J}_i =$ *Tröghetsmatris för kropp i.*
$\boldsymbol{v}_i =$ *Hastighet för masscentrum för kropp i.*
$\boldsymbol{u}_i =$ *Hastighet för kontaktpunkt för kropp i.*
$\boldsymbol{r}_i =$ *Kontaktpunktposition relativt masscentrum för kropp i.*
$\boldsymbol{\omega}_i =$ *Vinkelhastighet för kropp i.*
$\boldsymbol{p} =$ *Total impuls.*

På grund av antagande 1 räcker det att studera Δu_i d.v.s hur hastigheten för kontaktpunkterna på respektive kropp förändras, vilket görs med en kollisionsparameter γ. Vi låter $\gamma = p_z$ där p_z är den totala impulsen i normalriktningen, notera att initialt gäller $\gamma = p_z = 0$ och att den kontinuerligt ökar under kollisionen. Vid kollisionsögonblicket gäller att

$$\boldsymbol{u}_i = \boldsymbol{v}_i + \boldsymbol{\omega}_i \times \boldsymbol{r}_i. \tag{34}$$

och för förändringen vid γ.

$$\begin{aligned}
\Delta\boldsymbol{u}_i(\gamma) &= \boldsymbol{v}_i(\gamma) - \boldsymbol{v}_i(0) + \boldsymbol{\omega}_i(\gamma) \times \boldsymbol{r}_i(\gamma) - \boldsymbol{\omega}_i(0) \times \boldsymbol{r}_i(0) \\
&= \frac{1}{m_i}(\boldsymbol{p}_i(\gamma) - \boldsymbol{p}_i(0)) + \boldsymbol{J}_i^{-1}(\gamma)(\boldsymbol{r}_i \times \boldsymbol{p}_i(\gamma)) - \boldsymbol{J}_i^{-1}(0)(\boldsymbol{r}_i \times \boldsymbol{p}_i(0)) \\
&= \frac{1}{m_i}\boldsymbol{p}_i(\gamma) + \boldsymbol{J}_i^{-1}(\boldsymbol{r}_i \times \boldsymbol{p}_i(\gamma)) \\
&= (\frac{1}{m_i}\boldsymbol{I} - r_i^{\times}\boldsymbol{J}_i^{-1}r_i^{\times})\boldsymbol{p}_i(\gamma)
\end{aligned}$$

Där

$$\boldsymbol{r}_i^{\times} = \begin{bmatrix} 0 & -r_{iz} & r_{iy} \\ r_{iz} & 0 & r_{ix} \\ -r_{iy} & r_{ix} & 0 \end{bmatrix}$$

och notera att $\boldsymbol{J}_i$ och $\boldsymbol{r}_i$ enligt antagande 1 är konstanta och oberoende av γ, samt att $\boldsymbol{p}(0) = 0$. Vidare har vi

$$\begin{aligned}
\Delta\boldsymbol{u} &= \boldsymbol{u}_1 - \boldsymbol{u}_2 \\
&= (\frac{1}{m_1}\boldsymbol{I} - r_1^{\times}\boldsymbol{J}_1^{-1}r_1^{\times})\boldsymbol{p}_1(\gamma) - (\frac{1}{m_2}\boldsymbol{I} - r_2^{\times}\boldsymbol{J}_2^{-1}r_2^{\times})\boldsymbol{p}_2(\gamma) \\
&= M(\boldsymbol{p}_1 - \boldsymbol{p}_2) \\
&= M\Delta\boldsymbol{p}
\end{aligned}$$

Det gäller alltså att

$$\Delta\boldsymbol{u}' = M\boldsymbol{p}'$$

$$\Delta\boldsymbol{v}_i(\gamma) = \frac{1}{m}(\boldsymbol{p}(\gamma) - \boldsymbol{p}(0) = \frac{1}{m}\boldsymbol{p}(\gamma) \tag{35}$$

$$\Delta\boldsymbol{\omega}(\gamma) = \boldsymbol{J}^{-1}(\boldsymbol{r} \times \boldsymbol{p}(\gamma)). \tag{36}$$

Från antagande 3 inses att två fall behöver behandlas. Vi börjar med glidning. Låt $\theta(p_z) = arg(u_x + iu_y)$, alltså vinkeln för glidriktningen i kollisionskoordinat. Det gäller att

$$p_x' = \frac{dp_x}{dp_z} = \frac{dp_x}{dt}\frac{dt}{dp_z} = f_x\frac{1}{f_z}$$

och

$$p'_y = \frac{dp_y}{dp_z} = \frac{dp_y}{dt}\frac{dt}{dp_z} = f_y \frac{1}{f_z}.$$

$$p'_z = \frac{dp_z}{dp_z} = 1$$

Enligt antagande 3

$$f_x = -(\mu \cos\theta) f_z \Rightarrow p'_x = -\mu \cos\theta$$
$$f_y = -(\mu \sin\theta) f_z \Rightarrow p'_y = -\mu \sin\theta.$$

Alltså har vi

$$\begin{bmatrix} u'_x \\ u'_y \\ u'_z \end{bmatrix} = M \begin{bmatrix} -\mu \frac{u_x}{\sqrt{u_x^2+u_y^2}} \\ -\mu \frac{u_y}{\sqrt{u_x^2+u_y^2}} \\ 1 \end{bmatrix}. \tag{37}$$

I fallet för rullning gäller

$$u'_x = u'_y = 0 \Rightarrow \begin{bmatrix} 0 \\ 0 \\ u'_z \end{bmatrix} = M \begin{bmatrix} \alpha \\ \beta \\ 1 \end{bmatrix}$$

Enligt antagande måste det gälla att $|f_f|^2 = \alpha^2 + \beta^2 \leq \mu^2$.

□

B Superquadratics

Superquadratics är en samling av geometriska 3D-former med definierade huvudaxlar. Den implicita formen för superquadratics är:

$$|x|^r + |y|^s + |z|^t = 1$$

där $r, s, t \in \mathcal{R}_+$ bestämmer den geometriska formen. Vi kan även införa skalärer för varje respektive axel och får då för $A, B, C \in \mathcal{R}$:

$$\left|\frac{x}{A}\right|^r + \left|\frac{y}{B}\right|^s + \left|\frac{z}{C}\right|^t = 1.$$

vi parametrisera ekvationen med parametrarna u och v, där $-\frac{\pi}{2} \leq v \leq \frac{\pi}{2}$ och $-\pi \leq u < \pi$, och får då:

$$x(u,v) = Ag\left(v, \frac{2}{r}\right) g\left(u, \frac{2}{r}\right),$$
$$y(u,v) = Bg\left(v, \frac{2}{s}\right) f\left(u, \frac{2}{s}\right),$$
$$z(u,v) = Cf\left(v, \frac{2}{t}\right),$$

där f och g är våra axelfunktioner definierade som:

$$f(\omega, m) = \text{sgn}(\sin\omega)|\sin\omega|^m,$$
$$g(\omega, m) = \text{sgn}(\cos\omega)|\cos\omega|^m.$$

Genom teorin presenterad ovan så kan vi genera olika typer av formar som tillhör gruppen av Superquadratics [14].

C Kollsionsdetektion och restriktioner

Kollisionsdetektion

Genom att använda oss av två objekt i Blender så kan vi använda oss av radien på våra sfärer, vilket i detta fall är 30 cm, samt längden på planet för att upptäcka om det sker en kollision med en kollsionsmarginal på 0 mm, eftersom vi har en exakt beskrivning av ytan på sfären via radien. Om vi valt en annan form på våra konvexa objekt hade Blender approximerat ytan runt våra objekt, vilket gör att vi inte får en lika verklighetstrogen simulering.

Restriktioner

I första simuleringen tilldelar vi alla sfärer en massa på 4kg och sätter objekten 3m ifrån varandra. I simuleringen så har vi antagit en friktionskoefficient på 0.25. Nästa steg är att beräkna de krafter vi vill ska påverka våra objekt och vi har använt Newtons gravitationskraft $F = G\frac{m_1 m_2}{r^2}$. Vi kan approximera gravitationen på jorden till $m \cdot 9,80665$. Vi har dessutom använt två ytterligare krafter. En kraft $F_1 = 600$ N som påverkar sfär 1 i riktning mot sfär 2 och en kraft $F_2 = 600$ N som påverkar sfär 2 mot sfär 1. Krafterna avtar i enlighet med $1/r^F$. Genom att i simuleringen använda oss av en gravitation som flyttar objekten ner och samtidigt en kraft som flyttar objekten i respektive riktning så kommer vi naturligt få en rotation i i q_1- och q_2-riktningen.

Den andra simuleringen följer den första simuleringens premisser. Skillnaden från den första simuleringen är att kollisionen sker med ett plan. Planet är ett passivt objekt, vilket innebär att planet är fixerat för att kunna representeras som en vägg i verkligheten. Simuleringen utfördes så att det endast sker en kollision mellan planet och sfären. Orienteringen för sfären under simuleringen kan avläsas i figur 14.

D Visualiseringar

D.1 Blender

Kommandon och filer för visualiseringar som genomförts med Blender finns nedanför. Vi har valt att skriva med alla kommandon som gjorts i Blender och dessutom har vi laddat upp filerna på GitHub för att läsaren på ett smidigt sätt skall få tillgång till filerna.

Klicka här för att komma till GitHuB

Alternativt:
https://github.com/dunderlime/Visualisering-med-kvaternioner

D.1.1 Visualisering av figur: 6

```
bl_info = {
      "name": "Visualisering av Mobius-strip"
      "author" "Philip Karlsson",
      "version": (1,0),
      "blender": (2,80,0),

}

import bpy

bpy.ops.object.empty_add(type='ARROWS', location=(0, 0, 0))
bpy.context.object.rotation_mode = 'QUATERNION'
bpy.context.object.rotation_quaternion[0] = 0.707
bpy.context.object.rotation_quaternion[1] = 0
bpy.context.object.rotation_quaternion[2] = 0
bpy.context.object.rotation_quaternion[3] = 0.707

plane =bpy.ops.mesh.primitive_plane_add(location =(0, 0, 0))
bpy.context.object.scale[0] = 1
bpy.context.object.scale[1] = 8
bpy.context.object.scale[2] = 1
bpy.ops.object.editmode_toggle()

def view3d_find( return_area = False ):
    for area in bpy.context.window.screen.areas:
        if area.type == 'VIEW_3D':
            v3d = area.spaces[0]
            rv3d = v3d.region_3d
            for region in area.regions:
                if region.type == 'WINDOW':
                    if return_area: return region, rv3d, v3d, area
                    return region, rv3d, v3d
    return None, None

region, rv3d, v3d, area = view3d_find(True)

override = {
    'scene'  : bpy.context.scene,
    'region' : region,
    'area'   : area,
    'space'  : v3d
}

bpy.ops.mesh.loopcut_slide(
    override,
    MESH_OT_loopcut = {
        "number_cuts"           : 40,
        "smoothness"            : 0,
```

```
        "falloff"               : 'SMOOTH',
        "edge_index"            : 2,
        "object_index"          : 0,
        "mesh_select_mode_init" : (True, False, False)
    },
)

bpy.ops.object.editmode_toggle()
bpy.ops.object.modifier_add(type='SIMPLE_DEFORM')
bpy.context.object.modifiers["SimpleDeform"].deform_axis = 'Y'
bpy.context.object.modifiers["SimpleDeform"].angle = 3.14159
bpy.ops.object.modifier_apply(apply_as='DATA', modifier="SimpleDeform")
bpy.ops.object.modifier_add(type='SIMPLE_DEFORM')
bpy.context.object.modifiers["SimpleDeform"].deform_method = 'BEND'
bpy.context.object.modifiers["SimpleDeform"].origin = bpy.data.objects["Empty"]
bpy.context.object.modifiers["SimpleDeform"].deform_axis = 'Z'
bpy.context.object.modifiers["SimpleDeform"].angle = 6.28319
bpy.ops.object.modifier_apply(apply_as='DATA', modifier="SimpleDeform")

bpy.ops.object.editmode_toggle()
```

D.1.2 Visualisering av figur: 7-9

```
bl_info = {
      "name": "Visualisering av Balt-tricket"
      "author" "Philip Karlsson",
      "version": (1,0),
      "blender": (2,80,0),

}

import bpy

bpy.ops.object.empty_add(type='ARROWS', location=(0, 8, 0))
bpy.context.object.rotation_mode = 'QUATERNION'
bpy.context.object.rotation_quaternion[0] = 1
bpy.context.object.rotation_quaternion[1] = 0
bpy.context.object.rotation_quaternion[2] = 0
bpy.context.object.rotation_quaternion[3] = 0

plane =bpy.ops.mesh.primitive_plane_add(location =(0, 0, 0))
bpy.context.object.scale[0] = 1
bpy.context.object.scale[1] = 8
bpy.context.object.scale[2] = 1
bpy.ops.object.editmode_toggle()

def view3d_find( return_area = False ):
    for area in bpy.context.window.screen.areas:
        if area.type == 'VIEW_3D':
            v3d = area.spaces[0]
            rv3d = v3d.region_3d
            for region in area.regions:
                if region.type == 'WINDOW':
                    if return_area: return region, rv3d, v3d, area
                    return region, rv3d, v3d
    return None, None

region, rv3d, v3d, area = view3d_find(True)

override = {
    'scene'  : bpy.context.scene,
    'region' : region,
    'area'   : area,
    'space'  : v3d
}

bpy.ops.mesh.loopcut_slide(
    override,
    MESH_OT_loopcut = {
        "number_cuts"           : 40,
        "smoothness"            : 0,
```

```
        "falloff"               : 'SMOOTH',
        "edge_index"            : 2,
        "object_index"          : 0,
        "mesh_select_mode_init" : (True, False, False)
    },
)

bpy.ops.object.editmode_toggle()
bpy.ops.object.modifier_add(type='SIMPLE_DEFORM')
bpy.context.object.modifiers["SimpleDeform"].origin = bpy.data.objects["Empty"]
bpy.context.object.modifiers["SimpleDeform"].deform_axis = 'Y'
bpy.context.object.modifiers["SimpleDeform"].angle = 12.56637 # Change to 3.14159,
    6.28319, or 0 for other pictures
bpy.ops.object.modifier_apply(apply_as='DATA', modifier="SimpleDeform")
```

D.1.3 Visualisering av figur: 12

```
bl_info = {
    "name": "Visualisering av kvaternion Gauss map",
    "author": "Emil Hietanen",
    "version": (1, 0),
    "blender": (2, 80, 0),

}


import bpy

bpy.ops.mesh.primitive_uv_sphere_add(location=(0, 0, 0))
    bpy.ops.mesh.bisect(plane_co=(-0.390101, 0.169852, 0.854357), plane_no
    =(0.407088, 0.913326, -0.0107618), xstart=791, xend=793, ystart=588, yend=45)

bpy.ops.mesh.primitive_uv_sphere_add(location=(0, 0, 0))
  py.ops.mesh.bisect(plane_co=(-0.390101, 0.169852, 0.854357), plane_no=(0.407088,
    0.913326, -0.0107618), xstart=791, xend=793, ystart=588, yend=45)
  bpy.context.space_data.overlay.show_ortho_grid = True
  bpy.context.object.rotation_mode = 'QUATERNION'
  bpy.context.object.rotation_quaternion[0] = 0
  bpy.context.object.rotation_quaternion[1] = 1
  bpy.context.object.rotation_quaternion[2] = 0
  bpy.context.object.rotation_quaternion[3] = 0

bpy.context.space_data.shading.type = 'SOLID'
bpy.context.space_data.shading.type = 'WIREFRAME'
bpy.context.space_data.shading.type = 'WIREFRAME'
bpy.context.space_data.shading.type = 'WIREFRAME'
bpy.context.space_data.shading.type = 'SOLID'
bpy.context.space_data.shading.type = 'MATERIAL'
bpy.context.space_data.shading.type = 'RENDERED'


return {'FINISHED'}
```

D.1.4 Visulaisering av figur: 13

```
bl_info = {
    "name": "Fig: Kollisions detektion 1,
    "author": "Emil Hietanen",
    "version": (1, 0),
    "blender": (2, 80, 0),
}


import bpy
from bpy.types import Operator
from bpy.props import FloatVectorProperty
from bpy_extras.object_utils import AddObjectHelper, object_data_add
from mathutils import Vector

bpy.context.space_data.shading.type = 'WIREFRAME'
```

```
bpy.context.space_data.shading.type = 'WIREFRAME'
bpy.context.space_data.shading.type = 'WIREFRAME'
bpy.context.space_data.shading.type = 'SOLID'
bpy.context.space_data.shading.type = 'MATERIAL'
bpy.context.space_data.shading.type = 'RENDERED'
bpy.context.space_data.shading.type = 'SOLID'
bpy.context.space_data.shading.type = 'WIREFRAME'
bpy.context.space_data.overlay.show_overlays = False
bpy.context.space_data.overlay.show_overlays = True
bpy.context.space_data.shading.type = 'WIREFRAME'
bpy.context.space_data.shading.type = 'SOLID'
bpy.context.space_data.overlay.show_extra_edge_length = True
bpy.context.space_data.overlay.show_extra_edge_length = False
bpy.context.space_data.overlay.show_face_orientation = True
bpy.context.space_data.overlay.show_face_orientation = False
bpy.context.space_data.overlay.show_face_orientation = True
bpy.context.object.rotation_quaternion[1] = 10
bpy.context.object.rotation_quaternion[2] = 2
bpy.context.object.rotation_quaternion[3] = -4.6
bpy.ops.outliner.item_activate(extend=False, deselect_all=True)
bpy.context.object.rotation_mode = 'QUATERNION'
bpy.context.object.rotation_quaternion[2] = -26.4
bpy.context.object.rotation_quaternion[1] = -1
bpy.context.object.rotation_quaternion[2] = -13.2
bpy.ops.outliner.item_activate(extend=False, deselect_all=True)
bpy.ops.outliner.select_box(tweak=True, xmin=52, xmax=142, ymin=42, ymax=86)
bpy.ops.object.editmode_toggle()
bpy.ops.object.editmode_toggle()
bpy.context.space_data.shading.type = 'WIREFRAME'
bpy.context.space_data.shading.type = 'WIREFRAME'
bpy.context.space_data.shading.type = 'WIREFRAME'
bpy.context.space_data.shading.type = 'SOLID'
bpy.context.space_data.shading.type = 'WIREFRAME'
bpy.context.space_data.overlay.normals_length = 0.0301
bpy.context.space_data.show_gizmo = True
bpy.context.object.location[1] = -0.95
bpy.context.object.location[1] = -0.97
bpy.context.object.rotation_quaternion[1] = 10.6
bpy.context.object.location[1] = -0.93
bpy.context.object.location[1] = -0.92
bpy.context.object.location[1] = -0.91
bpy.context.object.location[1] = -0.9
bpy.context.space_data.show_gizmo = False
bpy.ops.object.editmode_toggle()
bpy.context.space_data.shading.type = 'WIREFRAME'
bpy.context.space_data.shading.type = 'WIREFRAME'
bpy.context.space_data.shading.type = 'SOLID'
bpy.context.space_data.shading.type = 'SOLID'
bpy.context.space_data.shading.type = 'WIREFRAME'
bpy.context.space_data.shading.type = 'SOLID'
bpy.context.space_data.shading.type = 'MATERIAL'
bpy.context.space_data.shading.type = 'WIREFRAME'
bpy.context.space_data.shading.type = 'SOLID'
bpy.context.space_data.shading.type = 'WIREFRAME'
bpy.ops.object.editmode_toggle()
bpy.context.space_data.shading.type = 'SOLID'
bpy.context.space_data.shading.type = 'WIREFRAME'
bpy.ops.object.editmode_toggle()
```

D.1.5 Visualisering av figur: 15

```
bl_info = {
    "name": "Kollision mellan sfar och plan",
    "author": "Philip Karlsson",
    "version": (1, 0),
    "blender": (2, 80, 0),

}



import bpy
import math
from mathutils import Quaternion

bpy.ops.mesh.primitive_plane_add(size=8, enter_editmode=False, location=(0, 0, 0))
bpy.context.space_data.context = 'PHYSICS'
bpy.ops.rigidbody.object_add()
bpy.context.object.rigid_body.type = 'PASSIVE'
bpy.context.object.rigid_body.friction = 0.25
bpy.context.object.rigid_body.restitution = 1
bpy.context.object.rigid_body.collision_shape = 'MESH'
bpy.ops.outliner.item_activate(extend=False, deselect_all=True)

bpy.ops.mesh.primitive_plane_add(size=3, enter_editmode=False, location=(0, 0, 0))
bpy.context.object.rotation_mode = 'QUATERNION'
bpy.context.object.rotation_quaternion[0] = 0.707
bpy.context.object.rotation_quaternion[1] = 0
bpy.context.object.rotation_quaternion[2] = 0.707
bpy.context.object.rotation_quaternion[3] = 0
bpy.context.space_data.context = 'PHYSICS'
bpy.ops.rigidbody.object_add()
bpy.context.object.rigid_body.type = 'PASSIVE'
bpy.context.object.rigid_body.collision_shape = 'MESH'
bpy.context.object.rigid_body.friction = 0.25
bpy.context.object.rigid_body.restitution = 0.3
bpy.context.object.rigid_body.use_margin = True
bpy.context.object.rigid_body.collision_margin = 0
bpy.context.space_data.context = 'MATERIAL'
bpy.ops.material.new()
bpy.context.object.active_material.use_nodes = False
bpy.context.object.active_material.diffuse_color = (0.0762984, 0.026208, 0.8, 1)
bpy.context.object.active_material.diffuse_color = (0.0762984, 0.026208, 0.8, 1)
bpy.ops.outliner.item_activate(extend=False, deselect_all=True)

bpy.ops.mesh.primitive_uv_sphere_add(radius=0.3, enter_editmode=False, location
    =(-3, 0, 1))
bpy.context.space_data.context = 'PHYSICS'
bpy.ops.rigidbody.object_add()
bpy.context.object.rigid_body.mass = 4
bpy.context.object.rigid_body.collision_shape = 'SPHERE'
bpy.context.object.rigid_body.friction = 0.25
bpy.context.object.rigid_body.restitution = 0.3
bpy.context.object.rigid_body.use_margin = True
bpy.context.object.rigid_body.collision_margin = 0
bpy.context.object.rigid_body.angular_damping = 0.848
bpy.context.space_data.context = 'MATERIAL'
bpy.ops.material.new()
bpy.context.object.active_material.use_nodes = False
bpy.context.object.active_material.diffuse_color = (0.8, 0.00778373, 0.0094196, 1)
bpy.context.object.active_material.diffuse_color = (0.8, 0.00778373, 0.0094196, 1)
bpy.ops.outliner.item_activate(extend=False, deselect_all=True)

bpy.ops.object.effector_add(type='WIND', enter_editmode=False, location=(0, 0, 0))
bpy.context.object.rotation_mode = 'QUATERNION'
bpy.context.object.rotation_quaternion[0] = 0.707
bpy.context.object.rotation_quaternion[1] = 0
bpy.context.object.rotation_quaternion[2] = 0.707
bpy.context.object.rotation_quaternion[3] = 0
```

```
bpy.context.object.field.falloff_type = 'TUBE'
bpy.context.object.field.use_min_distance = True
bpy.context.object.field.use_max_distance = True
bpy.context.object.field.distance_max = 2
bpy.context.object.field.strength = 600
bpy.ops.outliner.item_activate(extend=False, deselect_all=True)
bpy.ops.object.editmode_toggle()

# Go to animation tab to see the simulation
```

D.1.6 Visualisering av figur: 16

```
bl_info = {
    "name": "Comandos for Collision",
    "author": "Emil Hietanen",
    "version": (1, 0),
    "blender": (2, 80, 0),

}


import bpy
import math
from mathutils import Quaternion

bpy.ops.mesh.primitive_grid_add(size=2, enter_editmode=False, location=(0, 0, 0))
bpy.context.object.rotation_mode = 'QUATERNION'
bpy.context.object.scale[0] = 10
bpy.context.object.scale[1] = 10
bpy.context.object.scale[2] = 1
bpy.ops.outliner.item_activate(extend=False, deselect_all=True)



bpy.ops.mesh.primitive_uv_sphere_add(size=0.3, enter_editmode=False, location=(-3,
    0, 5))
bpy.ops.mesh.primitive_uv_sphere_add(size=0.3, enter_editmode=False, location=(3,
    0, 5))
bpy.context.object.rotation_mode = 'QUATERNION'
bpy.context.object.location[0] = 5
bpy.context.object.location[2] = 5
bpy.context.object.location[1] = 0
bpy.context.space_data.context = 'MATERIAL'
bpy.context.space_data.context = 'DATA'
bpy.context.space_data.context = 'PHYSICS'
bpy.ops.rigidbody.object_add()
bpy.context.object.rigid_body.collision_shape = 'SPHERE'
bpy.context.object.rigid_body.use_margin = True
bpy.context.object.rigid_body.collision_margin = 0
bpy.context.object.rigid_body.angular_damping = 0
bpy.ops.outliner.item_activate(extend=False, deselect_all=True)
bpy.ops.rigidbody.object_add()
bpy.context.object.rigid_body.type = 'PASSIVE'
bpy.ops.outliner.item_activate(extend=False, deselect_all=True)
bpy.context.space_data.context = 'OBJECT'
bpy.ops.outliner.item_activate(extend=False, deselect_all=True)
bpy.context.object.location[0] = -5
bpy.context.space_data.context = 'DATA'
bpy.context.space_data.context = 'PARTICLES'
bpy.context.space_data.context = 'MATERIAL'
bpy.context.space_data.context = 'DATA'
bpy.context.space_data.context = 'CONSTRAINT'
bpy.context.space_data.context = 'CONSTRAINT'
bpy.context.space_data.context = 'PHYSICS'
bpy.ops.outliner.item_activate(extend=False, deselect_all=True)
bpy.ops.object.effector_add(type='BOID', enter_editmode=False, location=(0, 0, 0))
bpy.ops.outliner.item_activate(extend=False, deselect_all=True)
bpy.context.object.field.strength = 100
bpy.ops.outliner.item_activate(extend=False, deselect_all=True)
bpy.ops.outliner.collection_delete()
bpy.ops.outliner.item_activate(extend=False, deselect_all=True)
```

```
bpy.context.scene.frame_start = 1
bpy.context.scene.use_preview_range = True
bpy.context.scene.use_preview_range = False
bpy.ops.object.effector_add(type='WIND', enter_editmode=False, location=(5, 0, 5))
bpy.context.object.field.falloff_type = 'TUBE'
bpy.context.object.field.strength = 600
bpy.context.object.field.use_min_distance = True
bpy.context.object.field.use_max_distance = True
bpy.context.object.field.distance_max = 5
bpy.context.object.field.use_absorption = True
bpy.context.object.field.use_absorption = False
bpy.context.space_data.context = 'OBJECT'

bpy.ops.object.effector_add(type='WIND', enter_editmode=False, location=(5-, 0, 5))
bpy.context.object.field.falloff_type = 'TUBE'
bpy.context.object.field.strength = 600
bpy.context.object.field.use_min_distance = True
bpy.context.object.field.use_max_distance = True
bpy.context.object.field.distance_max = 5
bpy.context.object.field.use_absorption = True
bpy.context.object.field.use_absorption = False
bpy.context.space_data.context = 'OBJECT'
bpy.context.object.rotation_mode = 'QUATERNION'
bpy.context.object.rotation_quaternion[2] = -0.707107
bpy.context.object.rotation_quaternion[0] = 0.707107
```

```
bpy.context.space_data.context = 'OBJECT'
bpy.context.space_data.context = 'DATA'
bpy.context.object.data.lens = 15
```

D.2 MatLab

Nedanför finns bifogad kod för de figurer som visualiserats i MatLab.

D.2.1 Visualisering av figur: 4 och 5

```
% Steriographic projection

% settup
dots = 100; % adjusts the resolution of the spheres
L = 3; % axis length
t = linspace(0,1,dots);
p0= [0 1]; %point_of_projection
n = 7; % number of projectionlines
StereoProj = @(x,y)x./(1-y); % progection function

%% Circle 2D -> 1D

%standard parametrisation of a circle
x = cos(2*pi*t);
y = sin(2*pi*t);

%projection onto x-axis
pX = StereoProj(x,y);

% some lines for enhanced visualization
theta = linspace(0,2*pi,n+1);


% ploting part
clf

hold on
axis([-L L -L/sqrt(1.6) L/sqrt(1.6)])

plot(x,y,'g')
```

```
plot(pX,0.*pX,'k-')

%plots projection lines
for i = 1:(length(theta)-1)
    projline = StereoProj(cos(pi+theta(i)),sin(pi+theta(i)));
    plot([p0(1) projline cos(pi+theta(i))],[p0(2) 0 sin(pi+theta(i))],'b')
end
plot(p0(1),p0(2),'r*')
shg

%% Sphere 3D -> 2D

% configure setup
dots = 20;
theta = linspace(0,2*pi,dots);
phi = linspace(0,pi,dots);
[THETA,PHI] = meshgrid(theta,phi);
pXY = @(x,z)x./(1-z);
n = 12;

% param sphere
X = cos(THETA).*sin(PHI);
Y = sin(THETA).*sin(PHI);
Z = cos(PHI);

p0 = [0 0 1];
Xp = pXY(X,Z);
Yp = pXY(Y,Z);
% ploting part
clf
hold on
% Note: to get the version 1 sphere remove the setings on facecolor and
%edgecolor below.
surf(Xp,Yp,0.*Xp,'facealpha',0.25,'facecolor','b')

colormap winter
surf(X,Y,Z,'facealpha',0.5,'edgecolor','none','facecolor','g')
plot3(p0(1),p0(2),p0(3),'r*')

% below we plots the lines between the projection and the normal object
for i = 1:n
    th = random('unif',0,2*pi);
    ph = random('unif',0,pi);

    x = cos(th).*sin(ph);
    y = sin(th).*sin(ph);
    z = cos(ph);

    plineX = pXY(x,z);
    plineY = pXY(y,z);
    plot3([p0(1) plineX x],[p0(2) plineY y],[p0(3) 0 z],'m')
end

axis([-L L -L L -L L])
view(-18,16)
shg

%% square root Method
% circle 2D -> 1D

%set up
th = linspace(0,2*pi); % angle for parametrisation
root = @(x,y) x; % this projection is so simple that tis function is redundant
n = 7;

% paramererize
x = cos(th);
y = sin(th);


```

```
% plot
clf
hold on
plot(x,y,'k')
plot([1 -1],[0 0],'g*') %plotsthe boundrypoints
plot(x,0.*x,'b-')% southernhemisphere projection
plot(x,0.*x+0.01,'r-') %plots northern hrmisphere projection,
% the shift of 0.01 in y is to make it visible due to the overlap

% plots projection lines
for i = 1:n
    x = random('unif',-1,1);
    y = random('unid',2);
    if(y==1)
       plot([x x], [sqrt(1-x^2) 0],'r--');
    else
        plot([x x], [-sqrt(1-x^2) 0],'b--');
    end
end
axis equal
shg

%% sphere 3D -> 2D
% set up
dots = 20;
theta = linspace(0,2*pi,dots);
phi = linspace(0,pi,dots);
[THETA,PHI] = meshgrid(theta,phi);
n = 12;

% param sphere
X = cos(THETA).*sin(PHI);
Y = sin(THETA).*sin(PHI);
Z = cos(PHI);

p0 = [0 0 1];
Xp = pXY(X,Z);
Yp = pXY(Y,Z);

% ploting part
clf
hold on
colormap winter

surf(X,Y,Z,'facealpha',0.15,'edgecolor','none'); %sphere
surf(X,Y,0.*Z,'facealpha',0.5,'facecolor','b') % southern hemisphere
surf(X,Y,0.*Z+0.01,'facealpha',0.5,'facecolor','r') % northern hemisphere

% plots projectionlines
for i = 1:n
    th = random('unif',0,2*pi);
    ph = random('unif',0,pi);

    x = cos(th).*sin(ph);
    y = sin(th).*sin(ph);
    z = cos(ph);

    if z>0
        plot3([x x],[y y],[z 0],'r-x')
    else
        plot3([x x],[y y],[z 0],'b-x')
    end
end
axis equal
view(3)
shg
```

D.2.2 Visualisering av figur: 10

```
% rolling ball

```

```
% settup
t0 = 1; %pause time in sek
dots = 100; % adjusts the resolution of the spheres
L = 3; % axis length
t = linspace(0,1,dots);

n = dots;
% configure setup
theta = linspace(0,2*pi,dots);
phi = linspace(0,pi,dots);
[THETA,PHI] = meshgrid(theta,phi);
[x,y] = meshgrid(linspace(-1,1,n),linspace(-1,1,n));
%% param sphere
X = cos(THETA).*sin(PHI);
Y = sin(THETA).*sin(PHI);
Z = cos(PHI);
C = Z; % to preserve collor in the sphere between rotations
CD = X;
% Rotmatrix

% ploting part 0
clf
hold on

colormap default
title('start')
surf(X,Y,Z,C,'facealpha',0.4,'edgecolor','none') % sphere
surf(X,Y,0.*Z,CD,'edgecolor','none') % Disk
%plot3([-1 1],[0 0],[0 0],'r')%x-axis
%plot3([0 0],[-1 1],[0 0],'b')%y-axis
%plot3([0 0],[0 0],[-1 1],'g')%z-axis
axis equal
axis off
%legend('','','x-axis','y','z')
view(-90,45)
shg
pause(t0)
%% rot 1 -pi/2 y-axis
clf
hold on

C = X;
CD = -Z;

title('Rotation kring -y-axel')
surf(X,Y,Z,C,'facealpha',0.4,'edgecolor','none') % sphere
surf(0.*X,Y,Z,CD,'edgecolor','none') % Disk
%plot3([-1 1],[0 0],[0 0],'r')%x-axis
%plot3([0 0],[-1 1],[0 0],'b')%y-axis
%plot3([0 0],[0 0],[-1 1],'g')%z-axis
axis equal
axis off
%legend('','','x-axis','y','z')
view(-90,45)
shg
pause(t0)
%% rot 2 pi/2 x-axis
clf
hold on

C = X;
CD = Y;

title('Rotation kring x-axel')
surf(X,Y,Z,C,'facealpha',0.4,'edgecolor','none') % sphere
surf(0.*X,Y,Z,CD,'edgecolor','none') % Disk
%plot3([-1 1],[0 0],[0 0],'r')%x-axis
%plot3([0 0],[-1 1],[0 0],'b')%y-axis
%plot3([0 0],[0 0],[-1 1],'g')%z-axis
axis equal
```

```
axis off
%legend('','','x-axis','y','z')
view(-90,45)
shg
pause(t0)
%% rot 2 pi/2 y-axis
C = Z;
CD = Y;

clf
hold on

colormap default
title('Rotation kring y-axel')
surf(X,Y,Z,C,'facealpha',0.4,'edgecolor','none') % sphere
surf(X,Y,0.*Z,CD,'edgecolor','none') % Disk
%plot3([-1 1],[0 0],[0 0],'r')%x-axis
%plot3([0 0],[-1 1],[0 0],'b')%y-axis
%plot3([0 0],[0 0],[-1 1],'g')%z-axis
axis equal
axis off
%legend('','','x-axis','y','z')
view(-90,45)
shg
```

D.2.3 Visualisering av figur: 11

```
%% Author: Hampus Ahlebrand
%Kod f r att visualisera kvaternionkurvan f r bolltricket
%d  handen rullar bollen med sm  cirkul ra r relser i xy-planet.
%Det antas h r att bollen har radien 1 vilket inneb r att l nden
%f r handens r relse i en viss riktning motsvarar rotationsvinkeln.


t=0; %motsvarar tid
T=0.1; %r relsen p g r i 0.1s
dt=0.0001; %Tidssteg
i=1; % index f r att spara komponenter i Q1,Q2 och Q2
theta=pi/250;%Rotationsvinkeln

dtheta=pi/50;%Riktningsf r ndring
dtheta_0=pi/50;

q=[1 0 0 0];%Initialv rdet, denna kvaternion svarar mot bollens
    begynnelseorientering

Q1=zeros(1,999);% De imagin ra komponenterna av de ber kande kvaternionerna
    sparas
Q2=zeros(1,999);% i Q1, Q2 och Q3.
Q3=zeros(1,999);


while t<T

    a=cos(theta/2);
    b=cos(dtheta);
    c=sin(theta/2);
    d=sin(dtheta);

    c1=1/sqrt(a^2+(c*d)^2+(c*b)^2); %normaliseringskonstant.

    qr=c1*[a c*d -c*b 0]; %rotationskvaternion

    q_0=qr(1)*q(1)-qr(2)*q(2)-qr(3)*q(3)-qr(4)*q(4); %kvaternionmult, qr*q
    q_1=qr(2)*q(1)+qr(1)*q(2)+qr(3)*q(4)-qr(4)*q(3);
    q_2=qr(3)*q(1)+qr(1)*q(3)+qr(4)*q(2)-qr(2)*q(4);
    q_3=qr(4)*q(1)+qr(1)*q(4)+qr(2)*q(3)-qr(3)*q(2);

    k=1/sqrt((q_0)^2+(q_1)^2+(q_2)^2+(q_3)^2);%normaliseringskonstant
    q=k*[q_0 q_1 q_2 q_3];

```

```
    Q1(i)=q(2);
    Q2(i)=q(3);
    Q3(i)=q(4);
    i=i+1;
    t=t+dt;
    dtheta=dtheta+dtheta_0;
end
plot3(Q1,Q2,Q3)%Resulterar i projektion av imagin ra delen mha kvadratmetoden
xlabel('q_1')
ylabel('q_2')
zlabel('q_3')
```

D.2.4 Visualisering av figur: 14

```
%% Author: Emil Hietanen
t=[3:-pi/90:0]%simuleringen bestod av length(t)frames
x=cos(t);
y=t/10000; %Eftersom vi vill ha y n ra noll f r att kunna plot3 och
%samtidigt ha det som i simulationen.
z_1=[3:-0.059:0];
z_2=[0.01:0.035:0.5];
z_3=0.51:-0.08:0;
z_4=[0.01 0.02 0.03 0.04 0.05 0.04 0.03 0.02 0.01 0 0.005 0.002 0];
z= [z_1 z_2 z_3 z_4]; %ger oss en vecktor med n gra av de punkter i q_3
plot3(x,y,z,'-x','MarkerIndices',86,'MarkerEdgeColor','r','Color','b','LineWidth'
    ,2)
xlabel('q_1')
ylabel('q_2')
zlabel('q_3')
grid on
hold on
xticks([-1 -0.5 0 0.5 1 1.5 2 2.5])
xticklabels({'-1','','','', '0'})
zticks([-1 -0.5 0 0.5 1 1.5 2 3])
zticklabels({'0','','','','','', '','1'})
yticks([-1 -0.5 0 0.5 1 1.5 2 3])
yticklabels({'0','','','','','', '','0'})
x=cos(t)
z=t/10000
plot3(x,y,z,'--','Color','b','LineWidth',2)
```